Superstore Customer Behavior Analysis: Predicting Response to Gold Membership Offer

2023-07-21

1. Project Details

1.1 Project Title

Superstore Customer Behavior Analysis: Predicting Response to Gold Membership Offer.

1.2. Background Information

The superstore wants to offer a special discount program called "Gold Membership" to its current customer base. The objective is to predict the likelihood of a customer's positive response to this offer. By understanding the key factors that influence this response, the superstore can optimize its marketing campaign, ensuring it is more effective and precisely targeted. This strategy will help the superstore reach the appropriate customer segment and increase the likelihood of a positive response to the gold membership offer.

1.3. Problem Statement

The objective of this analysis is to develop a predictive model that can effectively identify customers with a higher likelihood of purchasing the gold membership offer. By understanding the factors that influence customer response, the superstore can adjust their marketing strategy to target those customers who are more likely to say yes. This targeted strategy will enhance the conversion rate of customers into gold members and ultimately drive increased revenue for the superstore.

2. Reading the Data Source

For this project, we are utilizing data stored in a .csv file. Given our collaborative work environment and the use of GitHub for version control, we've chosen to host this file online to ensure all team members can access and load the data directly into R. The .csv file is hosted on Dropbox, on the following link:

Superstore Data Link

The dataset variables description is:

- Response (target) - 1 if customer accepted the offer in the last campaign, 0 otherwise
- ID - Unique ID of each customer
- Year_Birth - Age of the customer
- Complain - 1 if the customer complained in the last 2 years
- Dt_Customer - date of customer's enrollment with the company
- Education - customer's level of education

- Marital_Status - customer's marital status
- Kidhome - number of small children in customer's household
- Teenhome - number of teenagers in customer's household
- Income - customer's yearly household income
- MntFishProducts - the amount spent on fish products in the last 2 years
- MntMeatProducts - the amount spent on meat products in the last 2 years
- MntFruits - the amount spent on fruits products in the last 2 years
- MntSweetProducts - amount spent on sweet products in the last 2 years
- MntWines - the amount spent on wine products in the last 2 years
- MntGoldProds - the amount spent on gold products in the last 2 years
- NumDealsPurchases - number of purchases made with discount
- NumCatalogPurchases - number of purchases made using catalog (buying goods to be shipped through the mail)
- NumStorePurchases - number of purchases made directly in stores
- NumWebPurchases - number of purchases made through the company's website
- NumWebVisitsMonth - number of visits to company's website in the last month
- Recency - number of days since the last purchase

```
# Clean all variables
rm(list = ls())

# Verify that all variables have been removed
ls()
```

```
## character(0)
```

```
# Libraries Install
# List of packages to be installed and loaded
packages <- c("readr", "lubridate", "tidyverse", "ggplot2", "car", "corrgram",
              "RColorBrewer", "rpart", "rpart.plot", "randomForest", "caret",
              "ROCR", "rcompanion", "visreg", "MASS", "pscl", "dplyr", "pROC",
              "car", "ggplot2", "reshape2", "gridExtra")

# Install and load packages
for(package in packages){
  if(!require(package, character.only = TRUE)){
    install.packages(package)
    library(package, character.only = TRUE)
  }
}
```

```
## Loading required package: readr
```

```
## Loading required package: lubridate
```

```
##
## Attaching package: 'lubridate'
```

```
## The following objects are masked from 'package:base':
##
##     date, intersect, setdiff, union
```

```
## Loading required package: tidyverse

## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v dplyr   1.1.2     v stringr 1.5.0
## v forcats 1.0.0     v tibble  3.2.1
## v ggplot2 3.4.2     v tidyr   1.3.0
## v purrr   1.0.1
## -- Conflicts ---------------------------------------- tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors
## Loading required package: car
##
## Loading required package: carData
##
##
## Attaching package: 'car'
##
##
## The following object is masked from 'package:dplyr':
##
##     recode
##
##
## The following object is masked from 'package:purrr':
##
##     some
##
##
## Loading required package: corrgram
##
## Loading required package: RColorBrewer
##
## Loading required package: rpart
##
## Loading required package: rpart.plot
##
## Loading required package: randomForest
##
## randomForest 4.7-1.1
##
## Type rfNews() to see new features/changes/bug fixes.
##
##
## Attaching package: 'randomForest'
##
##
## The following object is masked from 'package:dplyr':
##
##     combine
##
##
## The following object is masked from 'package:ggplot2':
##
```

```
##     margin
##
##
## Loading required package: caret
##
## Loading required package: lattice
##
##
## Attaching package: 'lattice'
##
##
## The following object is masked from 'package:corrgram':
##
##     panel.fill
##
##
##
## Attaching package: 'caret'
##
##
## The following object is masked from 'package:purrr':
##
##     lift
##
##
## Loading required package: ROCR
##
## Loading required package: rcompanion
##
## Loading required package: visreg
##
## Loading required package: MASS
##
##
## Attaching package: 'MASS'
##
##
## The following object is masked from 'package:dplyr':
##
##     select
##
##
## Loading required package: pscl
##
## Classes and Methods for R developed in the
## Political Science Computational Laboratory
## Department of Political Science
## Stanford University
## Simon Jackman
## hurdle and zeroinfl functions by Achim Zeileis
##
## Loading required package: pROC
##
## Type 'citation("pROC")' for a citation.
```

```
##
##
## Attaching package: 'pROC'
##
##
## The following objects are masked from 'package:stats':
##
##     cov, smooth, var
##
##
## Loading required package: reshape2
##
##
## Attaching package: 'reshape2'
##
##
## The following object is masked from 'package:tidyr':
##
##     smiths
##
##
## Loading required package: gridExtra
##
##
## Attaching package: 'gridExtra'
##
##
## The following object is masked from 'package:randomForest':
##
##     combine
##
##
## The following object is masked from 'package:dplyr':
##
##     combine
```

```
# Read .csv file
url_location <- "https://www.dropbox.com/s/47xqmnl2464ee26/superstore_data.csv?dl=1"
superstore_data <- readr::read_csv(url_location)
```

```
## Rows: 2240 Columns: 22
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr  (3): Education, Marital_Status, Dt_Customer
## dbl (19): Id, Year_Birth, Income, Kidhome, Teenhome, Recency, MntWines, MntF...
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
head(superstore_data, 5)
```

```
## # A tibble: 5 x 22
##      Id Year_Birth Education  Marital_Status Income Kidhome Teenhome Dt_Customer
```

```
##   <dbl>      <dbl> <chr>      <chr>           <dbl>   <dbl>    <dbl> <chr>
## 1  1826       1970 Graduation Divorced        84835       0        0 6/16/2014
## 2     1       1961 Graduation Single          57091       0        0 6/15/2014
## 3 10476       1958 Graduation Married         67267       0        1 5/13/2014
## 4  1386       1967 Graduation Together        32474       1        1 11/5/2014
## 5  5371       1989 Graduation Single          21474       1        0 8/4/2014
## # i 14 more variables: Recency <dbl>, MntWines <dbl>, MntFruits <dbl>,
## #   MntMeatProducts <dbl>, MntFishProducts <dbl>, MntSweetProducts <dbl>,
## #   MntGoldProds <dbl>, NumDealsPurchases <dbl>, NumWebPurchases <dbl>,
## #   NumCatalogPurchases <dbl>, NumStorePurchases <dbl>,
## #   NumWebVisitsMonth <dbl>, Response <dbl>, Complain <dbl>
```

3. Data Cleaning and Preprocessing

During this phase we will prepare the dataset for analysis and modeling. First, the structure of the dataset will be inspected, in order to understand the structure and its features along with their data types. After that, missing values will be handled, either by imputing or deleting them. Finally, data transformation is applied to convert some variables to a more suited scale, and categorical variables need to be converted to dummy variables most likely, to be suited with the logistic regression algorithm.

3.1 Data Structure

We use str() to check the structure of our data. We will look into the data type of each column and the number of data points and variables.

```
# Obtain data structure
str(superstore_data)
```

```
## spc_tbl_ [2,240 x 22] (S3: spec_tbl_df/tbl_df/tbl/data.frame)
##  $ Id                 : num [1:2240] 1826 1 10476 1386 5371 ...
##  $ Year_Birth         : num [1:2240] 1970 1961 1958 1967 1989 ...
##  $ Education          : chr [1:2240] "Graduation" "Graduation" "Graduation" "Graduation" ...
##  $ Marital_Status     : chr [1:2240] "Divorced" "Single" "Married" "Together" ...
##  $ Income             : num [1:2240] 84835 57091 67267 32474 21474 ...
##  $ Kidhome            : num [1:2240] 0 0 0 1 1 0 0 0 0 0 ...
##  $ Teenhome           : num [1:2240] 0 0 1 1 0 0 0 1 1 1 ...
##  $ Dt_Customer        : chr [1:2240] "6/16/2014" "6/15/2014" "5/13/2014" "11/5/2014" ...
##  $ Recency            : num [1:2240] 0 0 0 0 0 0 0 0 0 0 ...
##  $ MntWines           : num [1:2240] 189 464 134 10 6 336 769 78 384 384 ...
##  $ MntFruits          : num [1:2240] 104 5 11 0 16 130 80 0 0 0 ...
##  $ MntMeatProducts    : num [1:2240] 379 64 59 1 24 411 252 11 102 102 ...
##  $ MntFishProducts    : num [1:2240] 111 7 15 0 11 240 15 0 21 21 ...
##  $ MntSweetProducts   : num [1:2240] 189 0 2 0 0 32 34 0 32 32 ...
##  $ MntGoldProds       : num [1:2240] 218 37 30 0 34 43 65 7 5 5 ...
##  $ NumDealsPurchases  : num [1:2240] 1 1 1 1 2 1 1 1 3 3 ...
##  $ NumWebPurchases    : num [1:2240] 4 7 3 1 3 4 10 2 6 6 ...
##  $ NumCatalogPurchases: num [1:2240] 4 3 2 0 1 7 10 1 2 2 ...
##  $ NumStorePurchases  : num [1:2240] 6 7 5 2 2 5 7 3 9 9 ...
##  $ NumWebVisitsMonth  : num [1:2240] 1 5 2 7 7 2 6 5 4 4 ...
##  $ Response           : num [1:2240] 1 1 0 0 1 1 1 0 0 0 ...
##  $ Complain           : num [1:2240] 0 0 0 0 0 0 0 0 0 0 ...
```

```
##  - attr(*, "spec")=
##   .. cols(
##   ..   Id = col_double(),
##   ..   Year_Birth = col_double(),
##   ..   Education = col_character(),
##   ..   Marital_Status = col_character(),
##   ..   Income = col_double(),
##   ..   Kidhome = col_double(),
##   ..   Teenhome = col_double(),
##   ..   Dt_Customer = col_character(),
##   ..   Recency = col_double(),
##   ..   MntWines = col_double(),
##   ..   MntFruits = col_double(),
##   ..   MntMeatProducts = col_double(),
##   ..   MntFishProducts = col_double(),
##   ..   MntSweetProducts = col_double(),
##   ..   MntGoldProds = col_double(),
##   ..   NumDealsPurchases = col_double(),
##   ..   NumWebPurchases = col_double(),
##   ..   NumCatalogPurchases = col_double(),
##   ..   NumStorePurchases = col_double(),
##   ..   NumWebVisitsMonth = col_double(),
##   ..   Response = col_double(),
##   ..   Complain = col_double()
##   .. )
##  - attr(*, "problems")=<externalptr>
```

Doing an initial analysis to the structure of the data, we can observe the following details:

- The dataset contains 2240 observations, with 22 distinct variables.
- The 'Dt_Customer' variable, denoting the enrollment date of each customer, is currently classified as a character variable. It would be more appropriate to transform this into a date variable.
- The variables 'Education' and 'Marital_Status', currently stored as character types, could be better represented as factor variables.
- The 'Year_Birth' variable could be restructured into age categories, to provide a more intuitive understanding of age distribution among customers.
- Interaction between 'Income' and 'NumStorePurchases': This interaction could capture the effect of a customer's income on their in-store purchasing behavior. Higher-income customers might make more in-store purchases.
- Interaction between 'Education' and 'Income': this interaction could capture the effect of a customer's education level on their income. Customers with higher levels of education might have higher incomes.
- Interaction between 'NumWebVisitsMonth' and 'NumWebPurchases': this interaction could capture the relationship between the number of website visits and the number of online purchases. Customers who visit the website more frequently might also make more online purchases.
- The individual spending amounts in various categories, such as 'MntWines', 'MntFruits', 'MntMeatProducts', etc., could be combined into a single cumulative expenditure variable. This could provide a view of each customer's total spending.

These are just some observations of things noticed while doing an initial analysis.

Check for Duplicate values

Let's investigate if there are any duplicate values.

```r
# Check for duplicate rows
duplicated_rows <- superstore_data %>% duplicated()

# Print the number of duplicated rows
print(paste("The number of duplicated rows is: ", sum(duplicated_rows)))
```

```
## [1] "The number of duplicated rows is:  0"
```

There are no duplicate values so we do not need to delete any rows based on that.

3.2. Handling Missing Values

Missing values need to be handled because most models do not handle them well. For missing values, the options are to:

- Substitute them with statistical estimates such as the mean, median, mode, etc.
- Eliminate them all together, if the amount of missing values is relatively low.

We will do some analysis to see if there are missing values in our dataset, and if so, how to handle them.

```r
# Get a statistical summary of each column in the dataset
summary_results <- summary(superstore_data)
print("Statistical summary of each column:")
```

```
## [1] "Statistical summary of each column:"
```

```r
print(summary_results)
```

```
##        Id          Year_Birth    Education         Marital_Status
##  Min.   :    0   Min.   :1893   Length:2240        Length:2240
##  1st Qu.: 2828   1st Qu.:1959   Class :character   Class :character
##  Median : 5458   Median :1970   Mode  :character   Mode  :character
##  Mean   : 5592   Mean   :1969
##  3rd Qu.: 8428   3rd Qu.:1977
##  Max.   :11191   Max.   :1996
##
##      Income          Kidhome          Teenhome      Dt_Customer
##  Min.   :  1730   Min.   :0.0000   Min.   :0.0000   Length:2240
##  1st Qu.: 35303   1st Qu.:0.0000   1st Qu.:0.0000   Class :character
##  Median : 51382   Median :0.0000   Median :0.0000   Mode  :character
##  Mean   : 52247   Mean   :0.4442   Mean   :0.5062
##  3rd Qu.: 68522   3rd Qu.:1.0000   3rd Qu.:1.0000
##  Max.   :666666   Max.   :2.0000   Max.   :2.0000
##  NA's   :24
##     Recency         MntWines         MntFruits     MntMeatProducts
##  Min.   : 0.00   Min.   :   0.00   Min.   :  0.0   Min.   :   0
##  1st Qu.:24.00   1st Qu.:  23.75   1st Qu.:  1.0   1st Qu.:  16
##  Median :49.00   Median : 173.50   Median :  8.0   Median :  67
##  Mean   :49.11   Mean   : 303.94   Mean   : 26.3   Mean   : 167
##  3rd Qu.:74.00   3rd Qu.: 504.25   3rd Qu.: 33.0   3rd Qu.: 232
```

```
##  Max.   :99.00   Max.   :1493.00   Max.   :199.0   Max.   :1725
##
##  MntFishProducts  MntSweetProducts  MntGoldProds    NumDealsPurchases
##  Min.   :  0.00   Min.   :  0.00   Min.   :  0.00   Min.   : 0.000
##  1st Qu.:  3.00   1st Qu.:  1.00   1st Qu.:  9.00   1st Qu.: 1.000
##  Median : 12.00   Median :  8.00   Median : 24.00   Median : 2.000
##  Mean   : 37.53   Mean   : 27.06   Mean   : 44.02   Mean   : 2.325
##  3rd Qu.: 50.00   3rd Qu.: 33.00   3rd Qu.: 56.00   3rd Qu.: 3.000
##  Max.   :259.00   Max.   :263.00   Max.   :362.00   Max.   :15.000
##
##  NumWebPurchases  NumCatalogPurchases NumStorePurchases NumWebVisitsMonth
##  Min.   : 0.000   Min.   : 0.000      Min.   : 0.00     Min.   : 0.000
##  1st Qu.: 2.000   1st Qu.: 0.000      1st Qu.: 3.00     1st Qu.: 3.000
##  Median : 4.000   Median : 2.000      Median : 5.00     Median : 6.000
##  Mean   : 4.085   Mean   : 2.662      Mean   : 5.79     Mean   : 5.317
##  3rd Qu.: 6.000   3rd Qu.: 4.000      3rd Qu.: 8.00     3rd Qu.: 7.000
##  Max.   :27.000   Max.   :28.000      Max.   :13.00     Max.   :20.000
##
##     Response         Complain
##  Min.   :0.0000   Min.   :0.000000
##  1st Qu.:0.0000   1st Qu.:0.000000
##  Median :0.0000   Median :0.000000
##  Mean   :0.1491   Mean   :0.009375
##  3rd Qu.:0.0000   3rd Qu.:0.000000
##  Max.   :1.0000   Max.   :1.000000
##
```

```
# Identify the exact location (row and column) of missing values
missing_values_locations <- which(is.na(superstore_data))
total_missing_values <- sum(is.na(superstore_data))

print(paste("Total number of missing values in the dataset: ", total_missing_values))
```

```
## [1] "Total number of missing values in the dataset:  24"
```

```
# List of columns that contain at least one missing value
listMissingColumns <- colnames(superstore_data)[ apply(superstore_data, 2, anyNA)]
print("Columns that contain at least one missing value: ")
```

```
## [1] "Columns that contain at least one missing value: "
```

```
print(listMissingColumns)
```

```
## [1] "Income"
```

```
# Identify the rows where missing values are located
missing_values_rows <- unique(row(superstore_data)[missing_values_locations])

# Print the row ID numbers where missing values are located
print("Row ID numbers where missing values are located:")
```

```
## [1] "Row ID numbers where missing values are located:"
```

```
print(missing_values_rows)
```

```
##  [1]  135  263  395  450  526  591  900  998 1097 1186 1214 1313 1516 1559 1694
## [16] 1805 1859 1864 1881 1968 1984 2140 2166 2171
```

```
# We are going to remove the rows with missing values, because the 'Income' variable has a
# wide range, making the mean or median not representative for imputation
if (!require(dplyr)) install.packages("dplyr")
library(dplyr)
superstore_data_filled <-superstore_data %>%
na.omit()
```

Looking at the results, we can see that there are 24 missing values in the Income column. Given that the 'Income' variable has a wide range, using the mean or median for imputation might not be representative. Therefore, we chose to remove the rows with missing values. The cleaned dataset is stored in 'superstore_data_no_missing_val'.

3.3. Data Transformation

Now during this step, we intend to modify the data to better suit our analysis and the predicting modeling building.

Transform 'Dt_Customer' from character to date type

```
# Transform 'Dt_Customer' column to a standardized date format (YYYY-MM-DD)
superstore_data_filled <- superstore_data_filled %>%
  mutate(Dt_Customer = mdy(Dt_Customer))

# Print the first few rows of the dataset to verify the transformation
head(superstore_data_filled,5)
```

```
## # A tibble: 5 x 22
##      Id Year_Birth Education  Marital_Status Income Kidhome Teenhome Dt_Customer
##   <dbl>      <dbl> <chr>      <chr>           <dbl>   <dbl>    <dbl> <date>
## 1  1826       1970 Graduation Divorced        84835       0        0 2014-06-16
## 2     1       1961 Graduation Single          57091       0        0 2014-06-15
## 3 10476       1958 Graduation Married         67267       0        1 2014-05-13
## 4  1386       1967 Graduation Together        32474       1        1 2014-11-05
## 5  5371       1989 Graduation Single          21474       1        0 2014-08-04
## # i 14 more variables: Recency <dbl>, MntWines <dbl>, MntFruits <dbl>,
## #   MntMeatProducts <dbl>, MntFishProducts <dbl>, MntSweetProducts <dbl>,
## #   MntGoldProds <dbl>, NumDealsPurchases <dbl>, NumWebPurchases <dbl>,
## #   NumCatalogPurchases <dbl>, NumStorePurchases <dbl>,
## #   NumWebVisitsMonth <dbl>, Response <dbl>, Complain <dbl>
```

```
str(superstore_data_filled)
```

```
## tibble [2,216 x 22] (S3: tbl_df/tbl/data.frame)
##  $ Id                 : num [1:2216] 1826 1 10476 1386 5371 ...
##  $ Year_Birth         : num [1:2216] 1970 1961 1958 1967 1989 ...
##  $ Education          : chr [1:2216] "Graduation" "Graduation" "Graduation" "Graduation" ...
```

```
##  $ Marital_Status     : chr [1:2216] "Divorced" "Single" "Married" "Together" ...
##  $ Income             : num [1:2216] 84835 57091 67267 32474 21474 ...
##  $ Kidhome            : num [1:2216] 0 0 0 1 1 0 0 0 0 0 ...
##  $ Teenhome           : num [1:2216] 0 0 1 1 0 0 0 1 1 1 ...
##  $ Dt_Customer        : Date[1:2216], format: "2014-06-16" "2014-06-15" ...
##  $ Recency            : num [1:2216] 0 0 0 0 0 0 0 0 0 0 ...
##  $ MntWines           : num [1:2216] 189 464 134 10 6 336 769 78 384 384 ...
##  $ MntFruits          : num [1:2216] 104 5 11 0 16 130 80 0 0 0 ...
##  $ MntMeatProducts    : num [1:2216] 379 64 59 1 24 411 252 11 102 102 ...
##  $ MntFishProducts    : num [1:2216] 111 7 15 0 11 240 15 0 21 21 ...
##  $ MntSweetProducts   : num [1:2216] 189 0 2 0 0 32 34 0 32 32 ...
##  $ MntGoldProds       : num [1:2216] 218 37 30 0 34 43 65 7 5 5 ...
##  $ NumDealsPurchases  : num [1:2216] 1 1 1 1 2 1 1 1 3 3 ...
##  $ NumWebPurchases    : num [1:2216] 4 7 3 1 3 4 10 2 6 6 ...
##  $ NumCatalogPurchases: num [1:2216] 4 3 2 0 1 7 10 1 2 2 ...
##  $ NumStorePurchases  : num [1:2216] 6 7 5 2 2 5 7 3 9 9 ...
##  $ NumWebVisitsMonth  : num [1:2216] 1 5 2 7 7 2 6 5 4 4 ...
##  $ Response           : num [1:2216] 1 1 0 0 1 1 1 0 0 0 ...
##  $ Complain           : num [1:2216] 0 0 0 0 0 0 0 0 0 0 ...
##  - attr(*, "na.action")= 'omit' Named int [1:24] 135 263 395 450 526 591 900 998 1097 1186 ...
##   ..- attr(*, "names")= chr [1:24] "135" "263" "395" "450" ...
```

The 'Dt_Customer' variable has been transformed to a date type column.

Drop column ID

Column ID does not provide any value, so it can be removed.

```
superstore_data_filled_noID <- superstore_data_filled %>% dplyr::select(-Id)

# Print the first few rows of the updated dataset
head(superstore_data_filled_noID, 5)
```

```
## # A tibble: 5 x 21
##   Year_Birth Education  Marital_Status Income Kidhome Teenhome Dt_Customer
##        <dbl> <chr>      <chr>           <dbl>   <dbl>    <dbl> <date>
## 1       1970 Graduation Divorced        84835       0        0 2014-06-16
## 2       1961 Graduation Single          57091       0        0 2014-06-15
## 3       1958 Graduation Married         67267       0        1 2014-05-13
## 4       1967 Graduation Together        32474       1        1 2014-11-05
## 5       1989 Graduation Single          21474       1        0 2014-08-04
## # i 14 more variables: Recency <dbl>, MntWines <dbl>, MntFruits <dbl>,
## #   MntMeatProducts <dbl>, MntFishProducts <dbl>, MntSweetProducts <dbl>,
## #   MntGoldProds <dbl>, NumDealsPurchases <dbl>, NumWebPurchases <dbl>,
## #   NumCatalogPurchases <dbl>, NumStorePurchases <dbl>,
## #   NumWebVisitsMonth <dbl>, Response <dbl>, Complain <dbl>
```

Aggregating Individual Expenditure Subcategories into a Total Expenditure variable

In the context of our analysis, the specific amounts spent on each subcategory (i.e., 'MntFishProducts', 'MntMeatProducts', 'MntFruits', 'MntSweetProducts', 'MntWines, MntGoldProds') are less relevant than the total amount spent. Therefore, to simplify our dataset and focus on the overall spending behavior, we will consolidate these subcategory expenditures into a single 'MntSpent' column. This new column will represent the total amount each customer has spent across all categories.

```r
# Aggregate sub spending amounts into a single amount and remove the individual sub amount
# columns
superstore_data_agg_spend <- superstore_data_filled_noID %>%
  mutate(MntTotalSpent = MntFishProducts + MntMeatProducts + MntFruits +
           MntSweetProducts + MntWines + MntGoldProds) %>%
  dplyr::select(-MntFishProducts, -MntMeatProducts, -MntFruits, -MntSweetProducts,
                -MntWines, -MntGoldProds)

head(superstore_data_agg_spend,5)
```

```
## # A tibble: 5 x 16
##   Year_Birth Education  Marital_Status Income Kidhome Teenhome Dt_Customer
##        <dbl> <chr>      <chr>           <dbl>   <dbl>    <dbl> <date>
## 1       1970 Graduation Divorced        84835       0        0 2014-06-16
## 2       1961 Graduation Single          57091       0        0 2014-06-15
## 3       1958 Graduation Married         67267       0        1 2014-05-13
## 4       1967 Graduation Together        32474       1        1 2014-11-05
## 5       1989 Graduation Single          21474       1        0 2014-08-04
## # i 9 more variables: Recency <dbl>, NumDealsPurchases <dbl>,
## #   NumWebPurchases <dbl>, NumCatalogPurchases <dbl>, NumStorePurchases <dbl>,
## #   NumWebVisitsMonth <dbl>, Response <dbl>, Complain <dbl>,
## #   MntTotalSpent <dbl>
```

Convert Education and Marital_Status to Categorical Variables

Next, we will assess the 'Education' and 'Marital_Status' columns. These columns contain categorical data, and it might be beneficial to convert them to factor variables in R.

```r
# Frequency count for 'Education' column
print("Frequency count for 'Education' column:")
```

```
## [1] "Frequency count for 'Education' column:"
```

```r
print(table(superstore_data_agg_spend$Education))
```

```
##
##   2n Cycle      Basic Graduation     Master        PhD
##        200         54       1116        365        481
```

```r
# Frequency count for 'Marital_Status' column
print("Frequency count for 'Marital_Status' column:")
```

```
## [1] "Frequency count for 'Marital_Status' column:"
```

```r
print(table(superstore_data_agg_spend$Marital_Status))
```

```
##
##   Absurd    Alone Divorced  Married   Single Together    Widow     YOLO
##        2        3      232      857      471      573       76        2
```

```r
# Remove rows with 'Absurd' and 'YOLO' as values
superstore_data_agg_spend <- superstore_data_agg_spend %>%
  filter(!(Marital_Status %in% c("Absurd", "YOLO", "Alone")))

# Convert 'Education' and 'Marital_Status' to factor variables
superstore_data_agg_spend$Education <- as.factor(superstore_data_agg_spend$Education)
superstore_data_agg_spend$Marital_Status <- as.factor(
  superstore_data_agg_spend$Marital_Status)

# Set 'Basic' and 'Single' as the reference levels for 'Education' and 'Marital_Status'
# respectively
superstore_data_agg_spend$Education <- relevel(superstore_data_agg_spend$Education,
                                               ref = 'Basic')
superstore_data_agg_spend$Marital_Status <- relevel(
  superstore_data_agg_spend$Marital_Status, ref = 'Single')

# Create dummy variables using ifelse()
superstore_data_categorical <- superstore_data_agg_spend %>%
  mutate(Together = ifelse(Marital_Status=="Together",1,0),
         Divorced = ifelse(Marital_Status=="Divorced",1,0),
         Widow = ifelse(Marital_Status=="Widow",1,0),
         Second_Cycle = ifelse(Education=="2n Cycle",1,0),
         Graduation = ifelse(Education=="Graduation",1,0),
         Master = ifelse(Education=="Master",1,0),
         PHD = ifelse(Education=="PhD",1,0))

str(superstore_data_categorical)
```

```
## tibble [2,209 x 23] (S3: tbl_df/tbl/data.frame)
##  $ Year_Birth         : num [1:2209] 1970 1961 1958 1967 1989 ...
##  $ Education          : Factor w/ 5 levels "Basic","2n Cycle",..: 3 3 3 3 3 5 2 3 5 5 ...
##  $ Marital_Status     : Factor w/ 5 levels "Single","Divorced",..: 2 1 3 4 1 1 3 4 3 3 ...
##  $ Income             : num [1:2209] 84835 57091 67267 32474 21474 ...
##  $ Kidhome            : num [1:2209] 0 0 0 1 1 0 0 0 0 0 ...
##  $ Teenhome           : num [1:2209] 0 0 1 1 0 0 0 1 1 1 ...
##  $ Dt_Customer        : Date[1:2209], format: "2014-06-16" "2014-06-15" ...
##  $ Recency            : num [1:2209] 0 0 0 0 0 0 0 0 0 0 ...
##  $ NumDealsPurchases  : num [1:2209] 1 1 1 1 2 1 1 1 3 3 ...
##  $ NumWebPurchases    : num [1:2209] 4 7 3 1 3 4 10 2 6 6 ...
##  $ NumCatalogPurchases: num [1:2209] 4 3 2 0 1 7 10 1 2 2 ...
##  $ NumStorePurchases  : num [1:2209] 6 7 5 2 2 5 7 3 9 9 ...
##  $ NumWebVisitsMonth  : num [1:2209] 1 5 2 7 7 2 6 5 4 4 ...
##  $ Response           : num [1:2209] 1 1 0 0 1 1 1 0 0 0 ...
##  $ Complain           : num [1:2209] 0 0 0 0 0 0 0 0 0 0 ...
##  $ MntTotalSpent      : num [1:2209] 1190 577 251 11 91 ...
##  $ Together           : num [1:2209] 0 0 0 1 0 0 0 1 0 0 ...
##  $ Divorced           : num [1:2209] 1 0 0 0 0 0 0 0 0 0 ...
##  $ Widow              : num [1:2209] 0 0 0 0 0 0 0 0 0 0 ...
##  $ Second_Cycle       : num [1:2209] 0 0 0 0 0 0 1 0 0 0 ...
##  $ Graduation         : num [1:2209] 1 1 1 1 1 0 0 1 0 0 ...
##  $ Master             : num [1:2209] 0 0 0 0 0 0 0 0 0 0 ...
##  $ PHD                : num [1:2209] 0 0 0 0 0 1 0 0 1 1 ...
##  - attr(*, "na.action")= 'omit' Named int [1:24] 135 263 395 450 526 591 900 998 1097 1186 ...
```

```
##   ..- attr(*, "names")= chr [1:24] "135" "263" "395" "450" ...
```

The values “Absurd”, “YOLO” and “Alone” in the ‘Marital_Status’ column seem to be either errors or non-standard entries, as they don’t correspond to typical categories for marital status, and there are only 2 of each for Absurd and YOLO, and 3 for Alone.

Since they are a very small proportion of the data, we can removed them.

We convert Education and Marital Status to categorical variables, with base = Basic and Single, respectively.

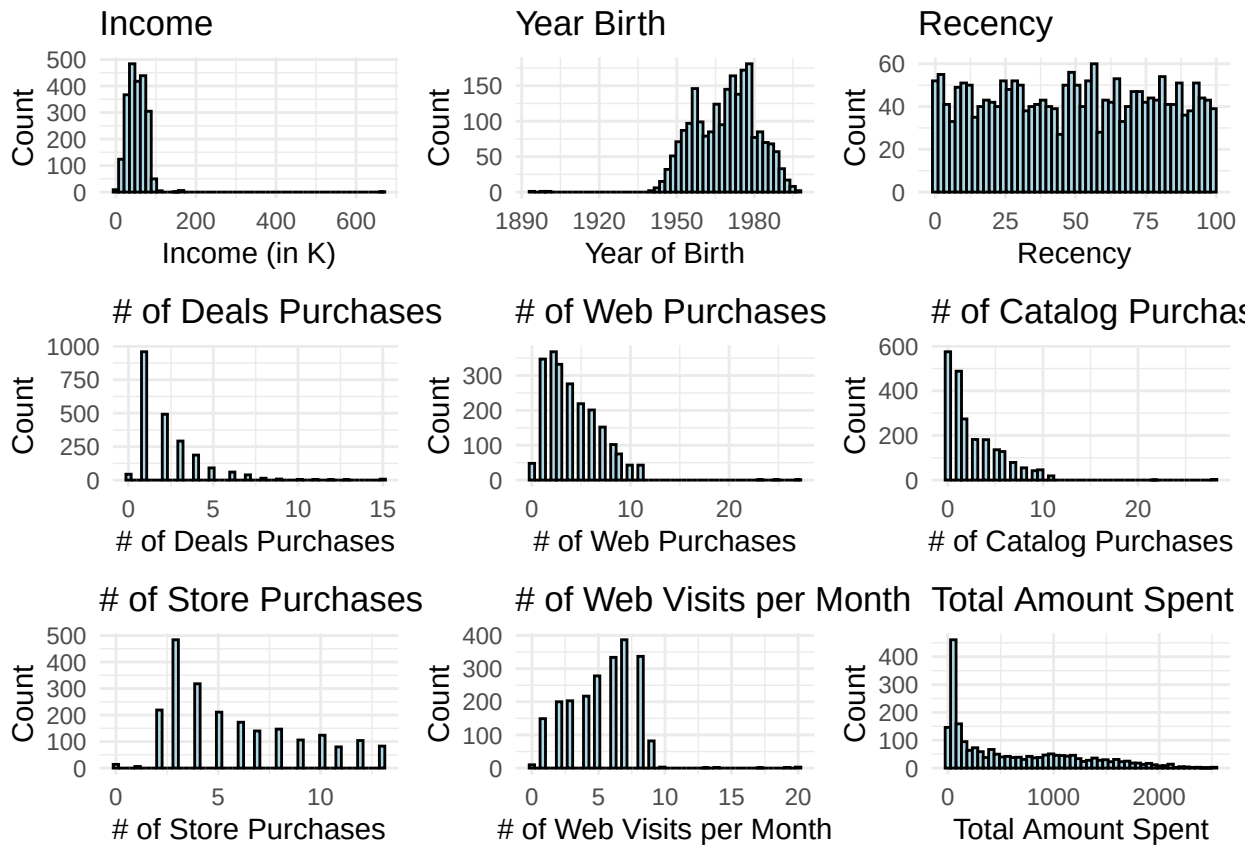
4. Exploratory Data Analysis

4.1. Histograms for Continuous Variables

In this section, we will explore our dataset by creating histograms for various variables.

```
# Histogram for Income
p1 <- ggplot(superstore_data_categorical, aes(x = Income/1000)) +
  geom_histogram(fill = 'lightblue', color = 'black', bins = 50) +
  theme_minimal() +
  labs(title = 'Income', x = 'Income (in K)', y = 'Count')

# Histogram for Year_Birth
p2 <- ggplot(superstore_data_categorical, aes(x = Year_Birth)) +
  geom_histogram(fill = 'lightblue', color = 'black', bins = 50) +
  theme_minimal() +
  labs(title = 'Year Birth', x = 'Year of Birth', y = 'Count')

# Histogram for Recency
p3 <- ggplot(superstore_data_categorical, aes(x = Recency)) +
  geom_histogram(fill = 'lightblue', color = 'black', bins = 50) +
  theme_minimal() +
  labs(title = 'Recency', x = 'Recency', y = 'Count')

# Histogram for NumDealsPurchases
p4 <- ggplot(superstore_data_categorical, aes(x = NumDealsPurchases)) +
  geom_histogram(fill = 'lightblue', color = 'black', bins = 50) +
  theme_minimal() +
  labs(title = '# of Deals Purchases', x = '# of Deals Purchases', y = 'Count')

# Histogram for NumWebPurchases
p5 <- ggplot(superstore_data_categorical, aes(x = NumWebPurchases)) +
  geom_histogram(fill = 'lightblue', color = 'black', bins = 50) +
  theme_minimal() +
  labs(title = '# of Web Purchases', x = '# of Web Purchases', y = 'Count')

# Histogram for NumCatalogPurchases
p6 <- ggplot(superstore_data_categorical, aes(x = NumCatalogPurchases)) +
  geom_histogram(fill = 'lightblue', color = 'black', bins = 50) +
  theme_minimal() +
  labs(title = '# of Catalog Purchases', x = '# of Catalog Purchases', y = 'Count')

# Histogram for NumStorePurchases
```

```r
p7 <- ggplot(superstore_data_categorical, aes(x = NumStorePurchases)) +
  geom_histogram(fill = 'lightblue', color = 'black', bins = 50) +
  theme_minimal() +
  labs(title = '# of Store Purchases', x = '# of Store Purchases', y = 'Count')

# Histogram for NumWebVisitsMonth
p8 <- ggplot(superstore_data_categorical, aes(x = NumWebVisitsMonth)) +
  geom_histogram(fill = 'lightblue', color = 'black', bins = 50) +
  theme_minimal() +
  labs(title = '# of Web Visits per Month', x = '# of Web Visits per Month', y = 'Count')

# Histogram for MntTotalSpent
p9 <- ggplot(superstore_data_categorical, aes(x = MntTotalSpent)) +
  geom_histogram(fill = 'lightblue', color = 'black', bins = 50) +
  theme_minimal() +
  labs(title = 'Total Amount Spent', x = 'Total Amount Spent', y = 'Count')

# Arrange the plots in a grid
grid.arrange(p1, p2, p3, p4, p5, p6, p7, p8, p9, ncol = 3)
```



From the histograms, we can make the following observations:

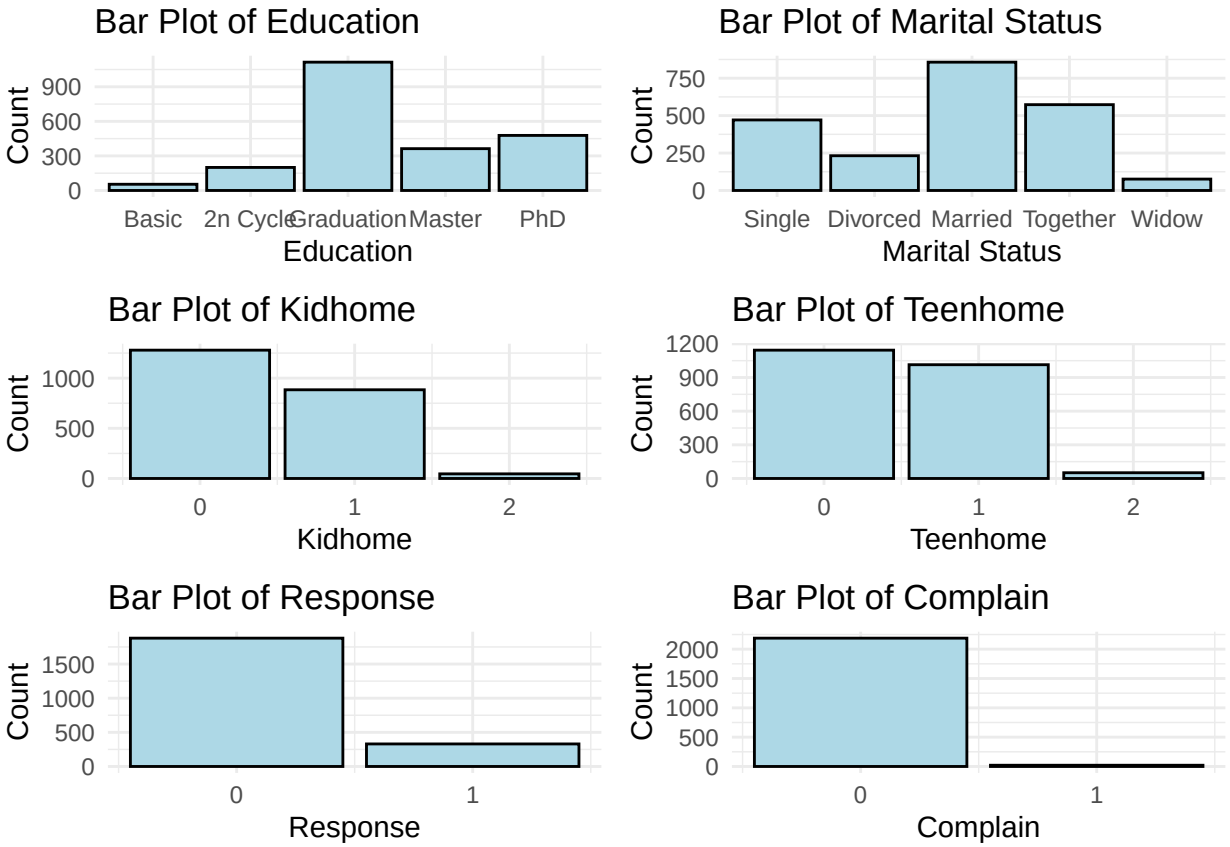
- **Income**: The majority of customers have an income between 0 and 100K. However, there are some outliers in the 150K range. These could be due to data entry errors or they could represent a small number of high-income customers. We will investigate this further in the outliers section.

- **Year of Birth**: Most of our customers are adults, as indicated by the concentration of birth years. Interestingly, there are some outliers born on or before 1900. These could be due to data entry errors or they could represent a small number of very old customers. We will investigate this further in the outliers section.

In the Outliers section, we will proceed to handle these outliers.

4.2. Bar Plot for Categorical Variables

```
# Create each plot
p1 <- ggplot(superstore_data_categorical, aes(x = Education)) +
  geom_bar(fill = 'lightblue', color = 'black') +
  theme_minimal() +
  labs(title = 'Bar Plot of Education', x = 'Education', y = 'Count')

p2 <- ggplot(superstore_data_categorical, aes(x = Marital_Status)) +
  geom_bar(fill = 'lightblue', color = 'black') +
  theme_minimal() +
  labs(title = 'Bar Plot of Marital Status', x = 'Marital Status', y = 'Count')

p3 <- ggplot(superstore_data_categorical, aes(x = Kidhome)) +
  geom_bar(fill = 'lightblue', color = 'black') +
  theme_minimal() +
  labs(title = 'Bar Plot of Kidhome', x = 'Kidhome', y = 'Count')

p4 <- ggplot(superstore_data_categorical, aes(x = Teenhome)) +
  geom_bar(fill = 'lightblue', color = 'black') +
  theme_minimal() +
  labs(title = 'Bar Plot of Teenhome', x = 'Teenhome', y = 'Count')

p5 <- ggplot(superstore_data_categorical, aes(x = Response)) +
  geom_bar(fill = 'lightblue', color = 'black') +
  scale_x_continuous(breaks = c(0, 1)) +
  theme_minimal() +
  labs(title = 'Bar Plot of Response', x = 'Response', y = 'Count')

p6 <- ggplot(superstore_data_categorical, aes(x = Complain)) +
  geom_bar(fill = 'lightblue', color = 'black') +
  scale_x_continuous(breaks = c(0, 1)) +
  theme_minimal() +
  labs(title = 'Bar Plot of Complain', x = 'Complain', y = 'Count')

# Arrange the plots in a grid
grid.arrange(p1, p2, p3, p4, p5, p6, ncol = 2)
```



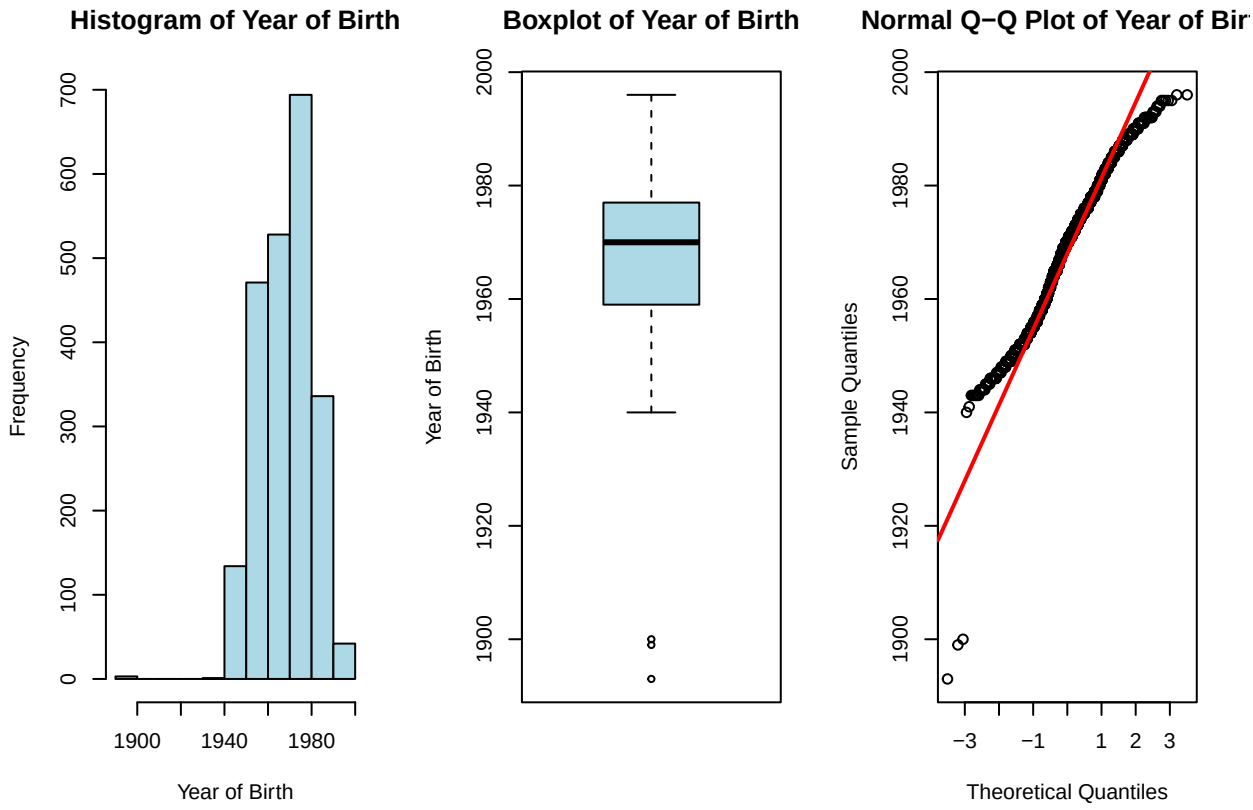
- From the 'Education' bar plot, we observe that the majority of our customers are graduates.
- Looking at the 'Marital_Status' bar plot, we see that most of our customers are either 'Married' or 'Together'.
- The 'Kidhome' and 'Teenhome' bar plots show an approximate even split between customers who do and do not have kids or teenagers at home. However, the number of customers with two kids or teenagers is minimal. This suggests that in the feature engineering stage, we could consider converting these variables to binary, with '0' representing no kids/teenagers and '1' representing one or more kids/teenagers.
- Finally, the 'Complain' and 'Response' bar plots provide some interesting insights. Despite the relatively low number of complaints, the majority of responses are negative.

4.3. Outliers

Now let's do some investigation on the outliers mentioned in the previous section.

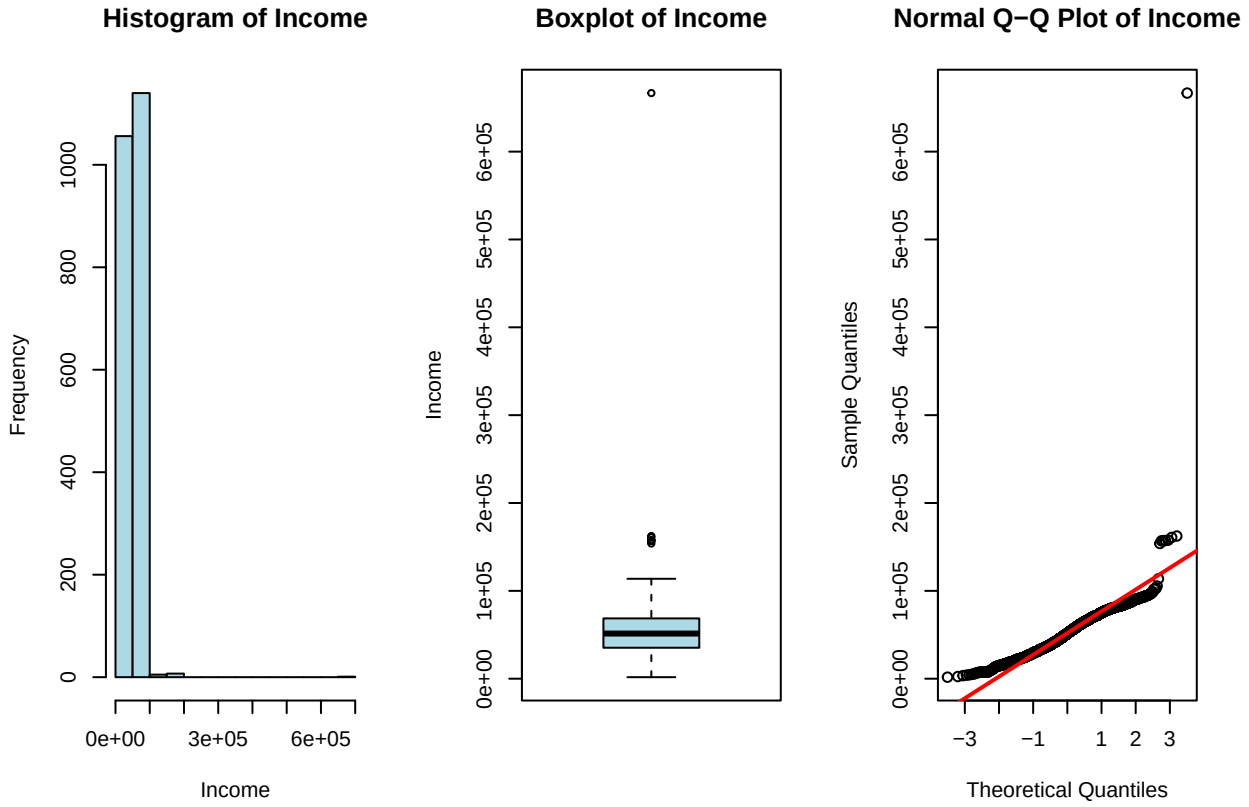
```
# Histogram, Q-Q plot, and boxplot for the Variable Year_Birth to detect possible Outliers
par(mfrow = c(1, 3))
# Histogram for 'Year_Birth'
hist(superstore_data_categorical$Year_Birth,
     main = 'Histogram of Year of Birth',
     xlab = 'Year of Birth',
     col = 'lightblue',
     border = 'black')

# Boxplot for 'Year_Birth'
```

```r
boxplot(superstore_data_categorical$Year_Birth,
        main = 'Boxplot of Year of Birth',
        ylab = 'Year of Birth',
        col = 'lightblue',
        border = 'black')

# Q-Q plot for 'Year_Birth'
qqnorm(superstore_data_categorical$Year_Birth,
       main = 'Normal Q-Q Plot of Year of Birth')
qqline(superstore_data_categorical$Year_Birth, col = "red", lwd = 2)
```

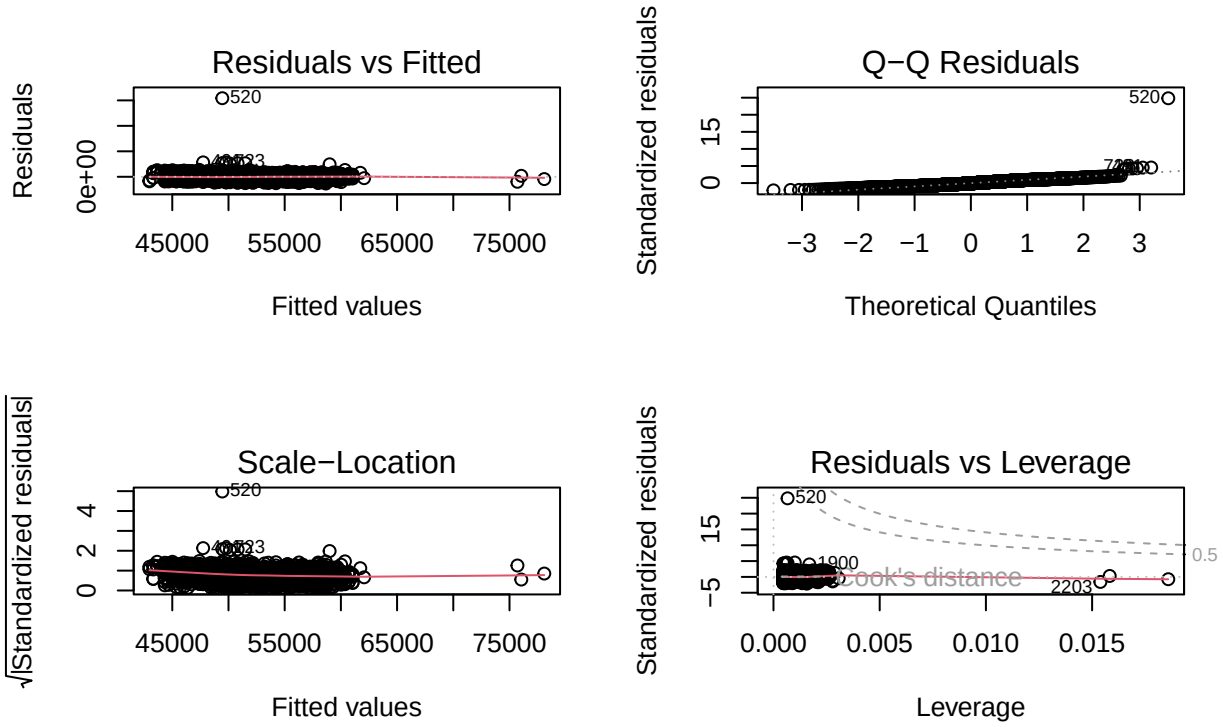



```r
par(mfrow = c(1, 1))

# Histogram, Q-Q plot, and boxplot for the Variable Income to detect possible Outliers
par(mfrow = c(1, 3))

# Histogram for 'Income'
hist(superstore_data_categorical$Income,
     main = 'Histogram of Income',
     xlab = 'Income',
     col = 'lightblue',
     border = 'black')

# Boxplot for 'Income'
boxplot(superstore_data_categorical$Income,
```

```
        main = 'Boxplot of Income',
        ylab = 'Income',
        col = 'lightblue',
        border = 'black')

# Q-Q plot for 'Income'
qqnorm(superstore_data_categorical$Income,
       main = 'Normal Q-Q Plot of Income')
qqline(superstore_data_categorical$Income, col = "red", lwd = 2)
```




```
# Calling a linear regression to remove Outliers using Cooks Distance method
a.lm <- lm(Income ~ Year_Birth, data = superstore_data_categorical)

# Diagnostic Plots for Linear Regression Model
par(mfrow = c(2, 2), oma = c(0, 0, 2, 0))
plot(a.lm, ask = FALSE)
```




```
par(mfrow = c(1, 1), oma = c(0, 0, 0, 0))

# Calculate Cook's distances
cooksD <- cooks.distance(a.lm)

# Identify influential observations
influential <- as.numeric(names(cooksD)[cooksD > 0.2] )
influential
```

```
## [1] 520
```

```
# Remove influential observations
superstore_no_outliers <- superstore_data_categorical[-influential,]

# Calculate mean and standard deviation of Year_Birth
mean = mean(superstore_no_outliers$Year_Birth)
std = sd(superstore_no_outliers$Year_Birth)

# Define thresholds for outliers
Tmin = mean-(3*std)
Tmax = mean+(3*std)

# Identify outliers
superstore_no_outliers$Year_Birth[which(superstore_no_outliers$Year_Birth < Tmin |
                                        superstore_no_outliers$Year_Birth > Tmax)]
```

```
## [1] 1893 1899 1900
```

```
# Remove outlier
superstore_no_outliers <- superstore_no_outliers %>%
  filter(!(Year_Birth %in% c(1893, 1899, 1900)))

str(superstore_no_outliers)
```

```
## tibble [2,205 x 23] (S3: tbl_df/tbl/data.frame)
##  $ Year_Birth         : num [1:2205] 1970 1961 1958 1967 1989 ...
##  $ Education          : Factor w/ 5 levels "Basic","2n Cycle",..: 3 3 3 3 3 5 2 3 5 5 ...
##  $ Marital_Status     : Factor w/ 5 levels "Single","Divorced",..: 2 1 3 4 1 1 3 4 3 3 ...
##  $ Income             : num [1:2205] 84835 57091 67267 32474 21474 ...
##  $ Kidhome            : num [1:2205] 0 0 0 1 1 0 0 0 0 0 ...
##  $ Teenhome           : num [1:2205] 0 0 1 1 0 0 0 1 1 1 ...
##  $ Dt_Customer        : Date[1:2205], format: "2014-06-16" "2014-06-15" ...
##  $ Recency            : num [1:2205] 0 0 0 0 0 0 0 0 0 0 ...
##  $ NumDealsPurchases  : num [1:2205] 1 1 1 1 2 1 1 1 3 3 ...
##  $ NumWebPurchases    : num [1:2205] 4 7 3 1 3 4 10 2 6 6 ...
##  $ NumCatalogPurchases: num [1:2205] 4 3 2 0 1 7 10 1 2 2 ...
##  $ NumStorePurchases  : num [1:2205] 6 7 5 2 2 5 7 3 9 9 ...
##  $ NumWebVisitsMonth  : num [1:2205] 1 5 2 7 7 2 6 5 4 4 ...
##  $ Response           : num [1:2205] 1 1 0 0 1 1 1 0 0 0 ...
##  $ Complain           : num [1:2205] 0 0 0 0 0 0 0 0 0 0 ...
##  $ MntTotalSpent      : num [1:2205] 1190 577 251 11 91 ...
##  $ Together           : num [1:2205] 0 0 0 1 0 0 0 1 0 0 ...
##  $ Divorced           : num [1:2205] 1 0 0 0 0 0 0 0 0 0 ...
##  $ Widow              : num [1:2205] 0 0 0 0 0 0 0 0 0 0 ...
##  $ Second_Cycle       : num [1:2205] 0 0 0 0 0 0 1 0 0 0 ...
##  $ Graduation         : num [1:2205] 1 1 1 1 1 0 0 1 0 0 ...
##  $ Master             : num [1:2205] 0 0 0 0 0 0 0 0 0 0 ...
##  $ PHD                : num [1:2205] 0 0 0 0 0 1 0 0 1 1 ...
##  - attr(*, "na.action")= 'omit' Named int [1:24] 135 263 395 450 526 591 900 998 1097 1186 ...
##   ..- attr(*, "names")= chr [1:24] "135" "263" "395" "450" ...
```

In our exploratory data analysis, we identified potential outliers in our dataset that may affect our subsequent analyses. Here, we detail our findings and the steps we took to address these issues.

Birth Year We observed that some users have a birth year recorded as 1900 or earlier. Given the context of our data, these entries are likely errors or placeholders and not representative of our actual customer base. To maintain the integrity of our data, we decided to remove these entries from our dataset.

Income The 'Income' variable showed a maximum value of 666,666, which is significantly higher than the mean income in our dataset. This extreme value could potentially skew our analyses and may not be representative of our typical customer. Therefore, we considered this as an outlier and removed it from our dataset as well.

Outlier Detection Methods

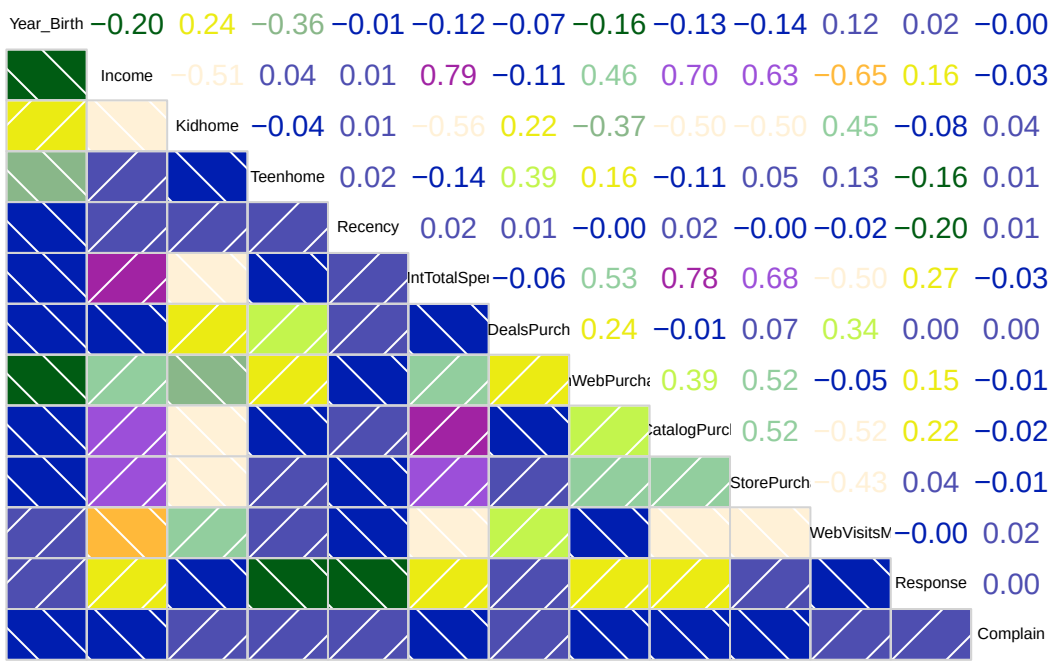
We employed two methods to detect and handle outliers in our dataset:

1. Cook's Distance: This method did not identify the outliers in 'Year_Birth' that we observed in our histograms, but it did identify the outlier in the Income variable.
2. Standard Deviation Method: Using the mean and standard deviation of 'Year_Birth', we identified three outlier values. We removed these entries from our dataset.

4.3. Correlation Matrix

In this section, we will create a correlation matrix to examine the relationships between different variables in our dataset.

```
color_func <- colorRampPalette(c("red", "orange", "white", "darkgreen", "blue", "yellow",
                                 "lightgreen", "purple", "brown"))

superstore_data_no_outliers_corr <- dplyr::select(superstore_no_outliers, Year_Birth,
                                          Income, Kidhome, Teenhome, Recency,
                                          MntTotalSpent, NumDealsPurchases,
                                          NumWebPurchases, NumCatalogPurchases,
                                          NumStorePurchases, NumWebVisitsMonth,
                                          Response, Complain)

suppressWarnings({
  corrgram(superstore_data_no_outliers_corr, order=FALSE,
           lower.panel=panel.shade,
           upper.panel=panel.cor,
           text.panel=panel.txt,
           col.regions = color_func)
})
```



The correlation matrix provides the following observations:

- There exists a significant positive correlation between the amount spent by a customer and their income, implying customers with higher income tend to spend more.

- Customers with kids at home appear to spend more, indicating a positive correlation between the presence of kids and the amount spent.
- Income and catalog purchases are positively correlated. This suggests that customers with higher income levels tend to make more catalog purchases.
- There is a negative correlation between having kids at home and both store purchases and catalog purchases. This may imply that customers with kids prefer the convenience of online shopping.
- High-income individuals seem to prefer not buying online, indicated by a negative correlation between income and online purchases.
- The probability of a positive response to the membership increases with income, demonstrating a positive correlation between income and membership acceptance.
- Finally, there is a positive response to catalog purchases. This suggests that customers who frequently purchase from catalogs are more likely to respond positively to the membership offer.

5. Feature Engineering

We are proceeding with additional data transformations to further optimize our dataset format for subsequent model utilization. Detailed below are the steps undertaken for this feature engineering phase.

Convert Kidhome and Teenhome variables to binary

```
num_kidhome_2 <- sum(superstore_no_outliers$Kidhome == 2)
print(paste("The number of entries with 2 kids at home is", num_kidhome_2))
```

```
## [1] "The number of entries with 2 kids at home is 46"
```

Our analysis shows that there are only 46 entries with households with two kids at home. Considering this limited representation, we have decided to streamline this variable into a binary form: households with kids and those without. By merging entries with one or two kids at home, we simplify our data representation without loosing important information.

```
# Convert Kidhome variable to binary
superstore_no_outliers$Kidhome_binary <- ifelse(superstore_no_outliers$Kidhome == 0, 0, 1)

# Remove the Kidhome column
superstore_no_outliers <- subset(superstore_no_outliers, select = -c(Kidhome))
```

```
num_teenhome_2 <- sum(superstore_no_outliers$Teenhome == 2)
print(paste("The number of entries with 2 teens at home is", num_teenhome_2))
```

```
## [1] "The number of entries with 2 teens at home is 51"
```

The scenario is the same for entries with teens at home, where only 51 instances are recorded. Consequently, we will unify the categories for households with either 1 or 2 teenagers, and convert this variable to binary.

```
# Convert Teenhome variable to binary
superstore_no_outliers$Teenhome_binary <- ifelse(superstore_no_outliers$Teenhome == 0, 0, 1)

# Remove the Teenhome column
superstore_no_outliers <- subset(superstore_no_outliers, select = -c(Teenhome))

# Print the modified dataset
print(superstore_no_outliers)
```

```
## # A tibble: 2,205 x 23
##    Year_Birth Education  Marital_Status Income Dt_Customer Recency
##         <dbl> <fct>      <fct>           <dbl> <date>        <dbl>
##  1       1970 Graduation Divorced        84835 2014-06-16        0
##  2       1961 Graduation Single          57091 2014-06-15        0
##  3       1958 Graduation Married         67267 2014-05-13        0
##  4       1967 Graduation Together        32474 2014-11-05        0
##  5       1989 Graduation Single          21474 2014-08-04        0
##  6       1958 PhD        Single          71691 2014-03-17        0
##  7       1954 2n Cycle   Married         63564 2014-01-29        0
##  8       1967 Graduation Together        44931 2014-01-18        0
##  9       1954 PhD        Married         65324 2014-11-01        0
## 10       1954 PhD        Married         65324 2014-11-01        0
## # i 2,195 more rows
## # i 17 more variables: NumDealsPurchases <dbl>, NumWebPurchases <dbl>,
## #   NumCatalogPurchases <dbl>, NumStorePurchases <dbl>,
## #   NumWebVisitsMonth <dbl>, Response <dbl>, Complain <dbl>,
## #   MntTotalSpent <dbl>, Together <dbl>, Divorced <dbl>, Widow <dbl>,
## #   Second_Cycle <dbl>, Graduation <dbl>, Master <dbl>, PHD <dbl>,
## #   Kidhome_binary <dbl>, Teenhome_binary <dbl>
```

Creation of Interaction Terms

Interaction terms allow us to capture the combined effect of two or more variables. We are including them with the purpose of understanding more the relationships within our dataset.

We are creating interaction terms between 'Income' and some purchase behavior indicators in order to explore how income and these purchasing behaviors impact the outcome variable.

Below are the interaction terms we are creating and their implications:

- Income_NumStore_Interact: this interaction term is the product of 'Income' and 'NumStorePurchases'. Its purpose is to examine whether the impact of income on the outcome changes based on the amount of store purchases. If significant, it could suggest that income level and store purchases are mutually dependent predictors.
- Income_NumWeb_Interact: similarly, this interaction term represents the combination of 'Income' and 'NumWebPurchases'. It allows us to see whether high-income customers tend to make more or fewer web purchases than would be predicted by the individual effects of income and web purchases alone.
- Income_MntTotal_Spent: this interaction term combines 'Income' and 'MntTotalSpent' (total amount spent). It may reveal whether customers with different income levels spend differently than would be expected based only on the separate effects of income and total spending.

In the next step, these interaction terms will be incorporated into our models.

```
# Create the interaction term between 'Income' and 'NumStorePurchases'
superstore_no_outliers$Income_NumStore_Interact <-
  superstore_no_outliers$Income * superstore_no_outliers$NumStorePurchases

# Create the interaction term between 'Income' and 'NumWebPurchases'
superstore_no_outliers$Income_NumWeb_Interact <-
  superstore_no_outliers$Income * superstore_no_outliers$NumWebPurchases

# Create the interaction term between 'Total_Purchases' and 'Income'
superstore_no_outliers$Income_MntTotal_Spent <-
```

```
  superstore_no_outliers$Income * superstore_no_outliers$MntTotalSpent

superstore_ready <- superstore_no_outliers

# Print the modified dataset
head(superstore_ready, 10)
```

```
## # A tibble: 10 x 26
##    Year_Birth Education  Marital_Status Income Dt_Customer Recency
##         <dbl> <fct>      <fct>           <dbl> <date>        <dbl>
##  1       1970 Graduation Divorced        84835 2014-06-16        0
##  2       1961 Graduation Single          57091 2014-06-15        0
##  3       1958 Graduation Married         67267 2014-05-13        0
##  4       1967 Graduation Together        32474 2014-11-05        0
##  5       1989 Graduation Single          21474 2014-08-04        0
##  6       1958 PhD        Single          71691 2014-03-17        0
##  7       1954 2n Cycle   Married         63564 2014-01-29        0
##  8       1967 Graduation Together        44931 2014-01-18        0
##  9       1954 PhD        Married         65324 2014-11-01        0
## 10       1954 PhD        Married         65324 2014-11-01        0
## # i 20 more variables: NumDealsPurchases <dbl>, NumWebPurchases <dbl>,
## #   NumCatalogPurchases <dbl>, NumStorePurchases <dbl>,
## #   NumWebVisitsMonth <dbl>, Response <dbl>, Complain <dbl>,
## #   MntTotalSpent <dbl>, Together <dbl>, Divorced <dbl>, Widow <dbl>,
## #   Second_Cycle <dbl>, Graduation <dbl>, Master <dbl>, PHD <dbl>,
## #   Kidhome_binary <dbl>, Teenhome_binary <dbl>,
## #   Income_NumStore_Interact <dbl>, Income_NumWeb_Interact <dbl>, ...
```

Conclusion of Data Preparation

With the completion of the data cleaning, preprocessing, exploratory analysis, and feature engineering steps, we have now a well-structured dataset, which is saved under the `superstore_ready` variable.

This refined dataset will serve as the foundation for our model building phase. We will proceed with the model development in the next stages of our analysis.

6. Model Building

Before starting to build the models, we will divide our dataset into two parts. 80% of the data will be designated as the training set, which will be utilized to train our models. The remaining 20%, which forms the testing set, will be employed to evaluate the performance and verify the accuracy of the models we develop.

```
# Total number of rows in dataset
n <- nrow(superstore_ready)
print(paste("The total number of rows in the data set is: ", n))
```

```
## [1] "The total number of rows in the data set is:  2205"
```

```r
# Number of rows for training set (80%)
n_train <- round(0.80 * n)

# Splitting the data into training
set.seed(123)
train_ind <- sample(1:n, n_train)

# Subset the dataset to training indices
train <- superstore_ready[train_ind, ]

# Excluding training indices to create the test set (20%)
test <- superstore_ready[-train_ind, ]

# Checking the size of the train and test dataset
print(paste("Training sample size: ", dim(train)[1]))
```

```
## [1] "Training sample size:  1764"
```

```r
print(paste("Testing sample size: ", dim(test)[1]))
```

```
## [1] "Testing sample size:  441"
```

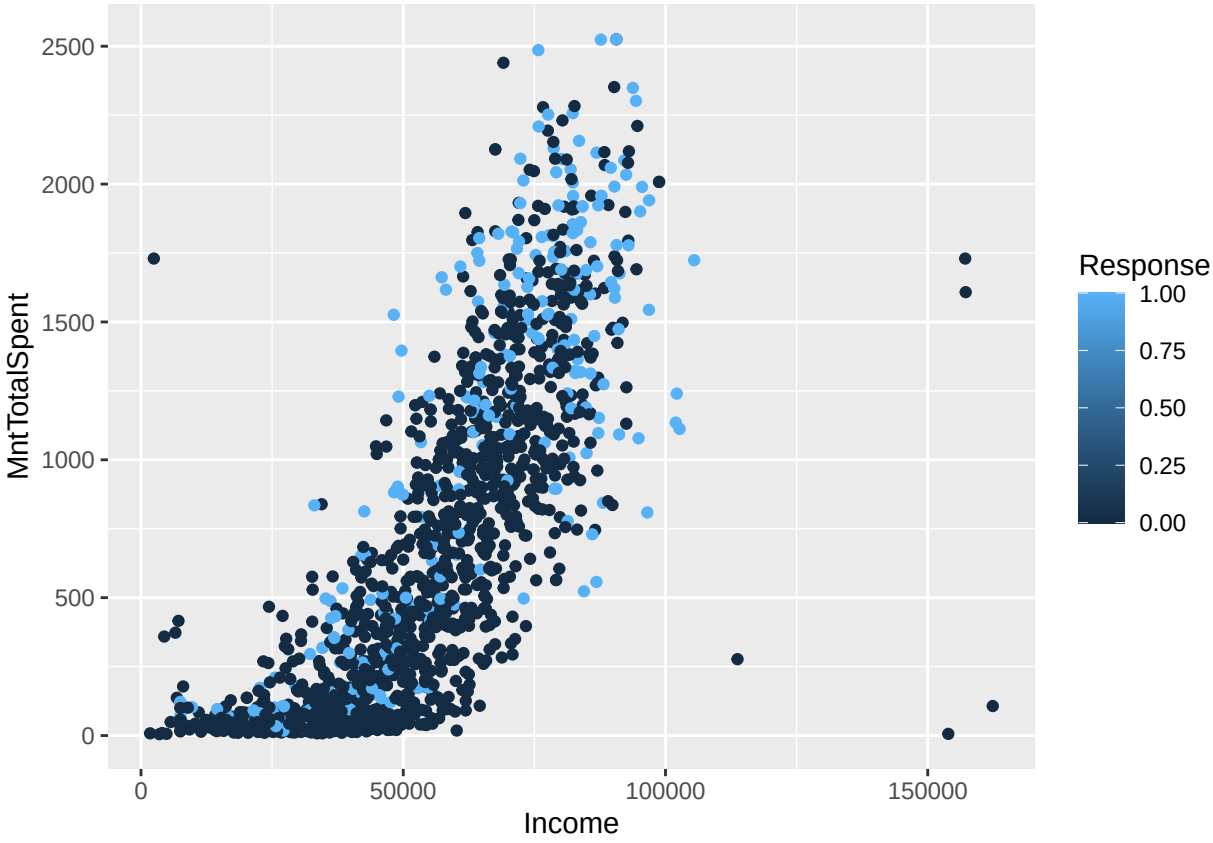
6.1. Logistic Regression

Now we will proceed to build a logistic regression model to predict the 'Response' variable. We will be using the training dataset to fit our model and the test dataset to evaluate its predictive performance.

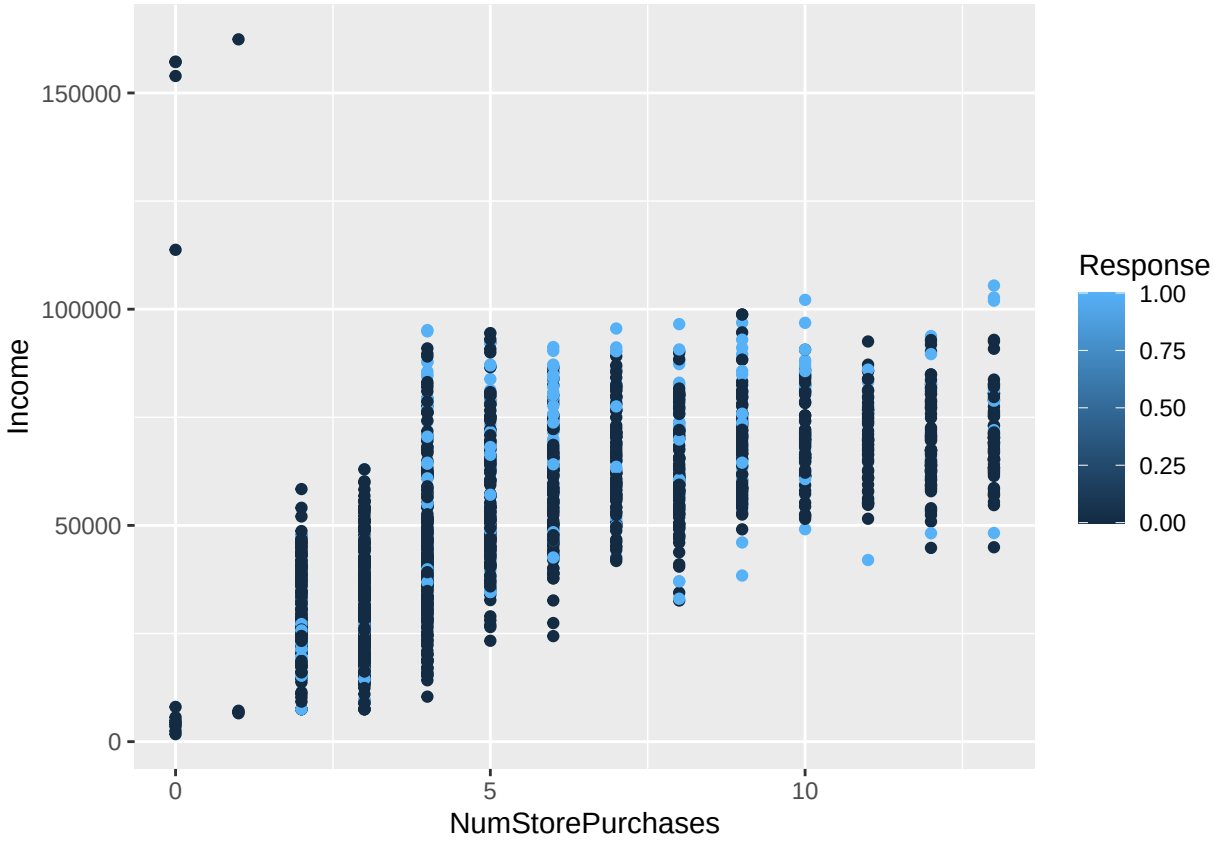
Base Model - Logistic Regression Model 1

We start by fitting a logistic regression model with all available predictors. This initial model will serve as our baseline, in order to compare subsequent models.

```r
par(mfrow = c(1, 2))
# Scatter Plot to see the data distribution per Income, Amount Total Spent and the Response
ggplot(train, aes(x=Income, y=MntTotalSpent, color=Response))+geom_point()
```




```
ggplot(train, aes(x=NumStorePurchases, y=Income, color=Response))+geom_point()
```




```
# Calling the Logistic Regression Model
glm1 <- glm( data=train , Response ~ Year_Birth + Second_Cycle + Graduation + Master +
               PHD + Together + Divorced + Widow + Income + Dt_Customer + Recency +
               NumDealsPurchases + NumWebPurchases + NumCatalogPurchases +
               NumStorePurchases + NumWebVisitsMonth + Complain + MntTotalSpent +
               Kidhome_binary + Teenhome_binary + Income_NumStore_Interact +
               Income_NumWeb_Interact + Income_MntTotal_Spent,  family="binomial")
summary(glm1)
```

```
##
## Call:
## glm(formula = Response ~ Year_Birth + Second_Cycle + Graduation +
##     Master + PHD + Together + Divorced + Widow + Income + Dt_Customer +
##     Recency + NumDealsPurchases + NumWebPurchases + NumCatalogPurchases +
##     NumStorePurchases + NumWebVisitsMonth + Complain + MntTotalSpent +
##     Kidhome_binary + Teenhome_binary + Income_NumStore_Interact +
##     Income_NumWeb_Interact + Income_MntTotal_Spent, family = "binomial",
##     data = train)
##
## Coefficients:
##                              Estimate Std. Error z value Pr(>|z|)
## (Intercept)                 2.498e+01  1.549e+01   1.612 0.106961
## Year_Birth                  4.292e-03  7.195e-03   0.597 0.550801
## Second_Cycle                1.166e+00  8.081e-01   1.442 0.149219
## Graduation                  1.468e+00  7.702e-01   1.906 0.056675 .
## Master                      1.612e+00  7.906e-01   2.039 0.041447 *
```

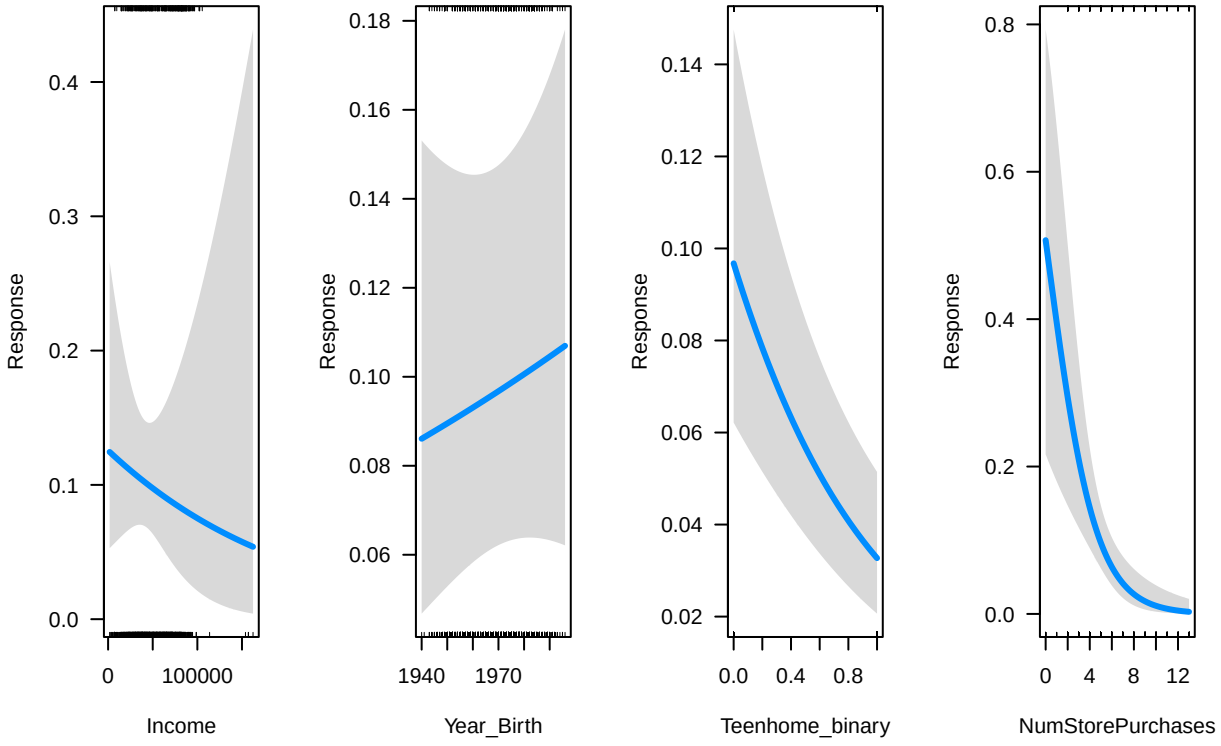
```
## PHD                        2.102e+00  7.836e-01   2.682 0.007310 **
## Together                  -4.343e-01  1.972e-01  -2.202 0.027651 *
## Divorced                   6.060e-01  2.324e-01   2.608 0.009112 **
## Widow                      7.504e-01  3.829e-01   1.960 0.050013 .
## Income                    -5.693e-06  1.086e-05  -0.524 0.600090
## Dt_Customer               -2.286e-03  3.730e-04  -6.128 8.89e-10 ***
## Recency                   -2.668e-02  2.884e-03  -9.250  < 2e-16 ***
## NumDealsPurchases          8.800e-02  4.865e-02   1.809 0.070493 .
## NumWebPurchases            1.134e-01  6.549e-02   1.731 0.083370 .
## NumCatalogPurchases        8.135e-02  4.017e-02   2.025 0.042884 *
## NumStorePurchases         -4.525e-01  1.235e-01  -3.664 0.000248 ***
## NumWebVisitsMonth          1.491e-01  4.869e-02   3.062 0.002195 **
## Complain                   2.862e-01  8.538e-01   0.335 0.737465
## MntTotalSpent              5.596e-04  7.217e-04   0.775 0.438067
## Kidhome_binary             1.474e-01  2.360e-01   0.624 0.532323
## Teenhome_binary           -1.153e+00  2.108e-01  -5.469 4.51e-08 ***
## Income_NumStore_Interact   3.830e-06  1.631e-06   2.349 0.018834 *
## Income_NumWeb_Interact    -1.707e-07  9.687e-07  -0.176 0.860100
## Income_MntTotal_Spent      8.921e-09  9.403e-09   0.949 0.342731
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 1537  on 1763  degrees of freedom
## Residual deviance: 1121  on 1740  degrees of freedom
## AIC: 1169
##
## Number of Fisher Scoring iterations: 6
```

```
# Pseudo R_squared values and Likelihood ratio test
nagelkerke(glm1)
```

```
## $Models
##
## Model: "glm, Response ~ Year_Birth + Second_Cycle + Graduation + Master + PHD + Together + Divorced +
## Null:  "glm, Response ~ 1, binomial, train"
##
## $Pseudo.R.squared.for.model.vs.null
##                              Pseudo.R.squared
## McFadden                             0.270695
## Cox and Snell (ML)                   0.210111
## Nagelkerke (Cragg and Uhler)         0.361263
##
## $Likelihood.ratio.test
##  Df.diff LogLik.diff  Chisq    p.value
##      -23     -208.03 416.06 8.7148e-74
##
## $Number.of.observations
##
## Model: 1764
## Null:  1764
##
## $Messages
```

```
## [1] "Note: For models fit with REML, these statistics are based on refitting with ML"
##
## $Warnings
## [1] "None"
```

```
# Plot of the Probabilities of Gold Membership
par( mfrow= c(1,4) )
      visreg(glm1, "Income", scale="response", rug=2, xlab="Income",
             ylab="Response")
      visreg(glm1, "Year_Birth", scale="response", rug=2, xlab="Year_Birth",
             ylab="Response")
      visreg(glm1, "Teenhome_binary", scale="response", rug=2, xlab="Teenhome_binary",
             ylab="Response")
      visreg(glm1, "NumStorePurchases", scale="response", rug=2, xlab="NumStorePurchases",
             ylab="Response")
```



From the Scatter plot of the distribution between Income, Amount Total Spent and Response, we can make the following observations:

1. Customers with higher income tend to spend more, as seen by the more concentrated dots in the top right corner of the plot. This suggests a positive correlation between income and total spending.
2. Both respondents and non-respondents to the Gold Membership campaign are distributed across all income and spending levels. The two groups are not distinctly separated, suggesting that income and total spending alone may not be strong predictors of response to the campaign.
3. There is a dense concentration of customers with relatively lower income and spending. This could be due to a larger number of customers in this income and spending bracket, or it could indicate that customers with lower income tend to spend less.

The McFadden Pseudo R-squared is an statistic used for comparing logistic regression models.It's similar to the R-squared value used in linear regression but with differences due to the fact that logistic regression is used for predicting binary outcomes rather than continuous variables.

In logistic regression, Pseudo R-squared values are more useful for comparing the fit of different models with the same outcome variable, rather than interpreting the goodness of fit of a single model.

After running the logistic regression model, the McFadden Pseudo R-squared value is approximately 0.27. This means that approximately 27% of the variability in the response variable is explained by the predictor variables included in the model. This can be interpreted as indicating that the predictor variables in the model provide a significant amount of information about the outcome, but there is still an amount of variability left unexplained.

Now that we had fit our base binary logistic regression model, we need to check how well the fitted model performs on the 20% test data.

```
# Predict the test dataset in the Base Model
pred <- predict(glm1, test, type="response")
predicted <- round(pred) # round of the value; >0.5 will convert to
                         # 1 else 0

# Creating a Confusion Matrix
tab <- table(Predicted = predicted, Reference = test$Response)

print("Confusion Matrix")
```

```
## [1] "Confusion Matrix"
```

```
print(tab, quote = FALSE)
```

```
##           Reference
## Predicted   0   1
##         0 383  36
##         1   6  16
```

```
# Calculate accuracy
accuracy <- sum(diag(tab)) / sum(tab)

# Print accuracy
print(paste("Accuracy:", accuracy))
```

```
## [1] "Accuracy: 0.904761904761905"
```

This matrix shows true negatives (383), false positives (36), false negatives (6), and true positives (16).

The provided accuracy result is 0.904761904761905, or approximately 90.48%. This means that the model achieved an accuracy of around 90.48% on the test dataset.

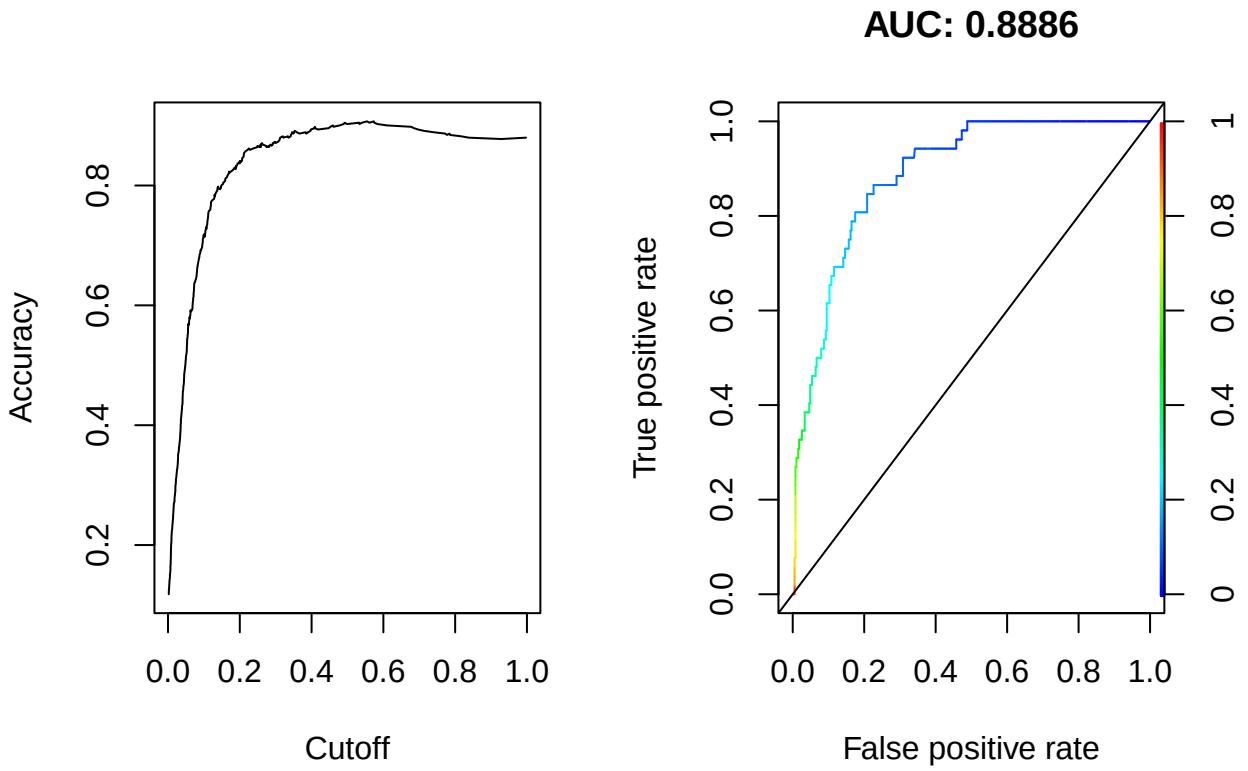
Now let's calculate the base logistic regression model performance using ROC.

```
par(mfrow = c(1, 2))

# Create a prediction object for ROC analysis
pred.rocr <- prediction(pred, test$Response)
```

```r
# Evaluate the performance of the predictions using accuracy
eval <- performance(pred.rocr, "acc")

# Plot the accuracy curve
plot(eval)

# Identify the best cutoff value that maximizes accuracy
max <- which.max(slot(eval, "y.values")[[1]])
acc <- slot(eval, "y.values")[[1]][max]   # Accuracy values
cut <- slot(eval, "x.values")[[1]][max]   # Cutoff values

# Print the accuracy and cutoff value
print(c(Accuracy = acc, Cutoff = cut))
```

```
##  Accuracy Cutoff.37
## 0.9070295 0.5732952
```

```r
# Calculate the Area Under the Curve (AUC)
perf_rocr <- performance(pred.rocr, measure = "auc", x.measure = "cutoff")
perf_rocr@y.values[[1]] <- round(perf_rocr@y.values[[1]], digits = 4)

# Create a performance object for true positive rate (TPR) and false positive rate (FPR)
perf.tpr.fpr.rocr <- performance(pred.rocr, "tpr", "fpr")

# Plot the ROC curve with colorization and AUC value
plot(perf.tpr.fpr.rocr, colorize = TRUE, main = paste("AUC:", (perf_rocr@y.values)))
abline(a = 0, b = 1)
```



Going over these results:

- Accuracy: 0.9070295, or approximately 90.70%, which indicates the proportion of correct predictions made by the model.
- Cutoff: 0.5732952. Predicted probabilities above this cutoff are classified as positive outcomes, while those below the cutoff are classified as negative outcomes.
- AUC: 0.8886. The Area Under the Curve (AUC) value of 0.8886 suggests that the model has good performance in distinguishing between positive and negative outcomes.

The high accuracy and AUC values indicate that the model is performing well in its predictions.

After obtaining our baseline model, we proceed with a model selection process. This involves iterative testing, comparing, and refining the model to improve the predictive accuracy, while also keeping the model as simple as possible. This balance between model accuracy and complexity is known as the bias-variance trade off.

Model Refinement - Logistic Regression Model 2

To refine our model, we employ a backward stepwise variable selection process based on the AIC (Akaike Information Criterion). This method starts with the full model and removes predictors one at a time, based on their significance, until no further improvement (decrease) in the AIC is observed.

```
# Perform stepwise selection
glm2 <- stepAIC(glm1, direction = "backward", trace = FALSE) # model is your initial model
summary(glm2)
```

```
##
```

```
## Call:
## glm(formula = Response ~ Second_Cycle + Graduation + Master +
##     PHD + Together + Divorced + Widow + Dt_Customer + Recency +
##     NumDealsPurchases + NumWebPurchases + NumCatalogPurchases +
##     NumStorePurchases + NumWebVisitsMonth + Teenhome_binary +
##     Income_NumStore_Interact + Income_MntTotal_Spent, family = "binomial",
##     data = train)
##
## Coefficients:
##                             Estimate Std. Error z value Pr(>|z|)
## (Intercept)                3.377e+01  5.996e+00   5.633 1.77e-08 ***
## Second_Cycle               1.124e+00  7.994e-01   1.406 0.159705
## Graduation                 1.421e+00  7.585e-01   1.873 0.061042 .
## Master                     1.562e+00  7.766e-01   2.012 0.044264 *
## PHD                        2.037e+00  7.679e-01   2.653 0.007977 **
## Together                  -4.395e-01  1.969e-01  -2.232 0.025646 *
## Divorced                   5.805e-01  2.298e-01   2.526 0.011529 *
## Widow                      6.866e-01  3.747e-01   1.832 0.066904 .
## Dt_Customer               -2.319e-03  3.705e-04  -6.258 3.90e-10 ***
## Recency                   -2.661e-02  2.879e-03  -9.244  < 2e-16 ***
## NumDealsPurchases          9.871e-02  4.497e-02   2.195 0.028144 *
## NumWebPurchases            1.001e-01  3.079e-02   3.252 0.001147 **
## NumCatalogPurchases        8.044e-02  3.679e-02   2.187 0.028776 *
## NumStorePurchases         -3.834e-01  9.883e-02  -3.879 0.000105 ***
## NumWebVisitsMonth          1.674e-01  4.277e-02   3.914 9.09e-05 ***
## Teenhome_binary           -1.247e+00  1.928e-01  -6.470 9.81e-11 ***
## Income_NumStore_Interact   2.858e-06  1.317e-06   2.170 0.029986 *
## Income_MntTotal_Spent      1.436e-08  2.879e-09   4.988 6.10e-07 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 1537.0  on 1763  degrees of freedom
## Residual deviance: 1123.2  on 1746  degrees of freedom
## AIC: 1159.2
##
## Number of Fisher Scoring iterations: 6
```

The resulting model includes predictors that were selected or removed based on their significance.

Predictors with smaller p-values are considered more significant and have a stronger association with the response variable. The coefficients associated with each predictor represent their estimated effects on the log-odds of the response variable.

The selected predictors for the refined model are as follows:

Second_Cycle, Graduation, Master, PHD, Together, Divorced, Widow, Dt_Customer, Recency, NumDealsPurchases, NumWebPurchases, NumCatalogPurchases, NumStorePurchases, NumWebVisitsMonth, Teenhome_binary, Income_NumStore_Interact, Income_MntTotal_Spent

Overall, this refined model with the selected variables enhances the predictive capabilities of the logistic regression model.

Now let's predict the Response on the test set:

```r
# Predict the test dataset in Model 2
pred2 <- predict(glm2, test, type="response")
predicted2 <- round(pred2) # round of the value; >0.5 will convert to
                           # 1 else 0

# Creating a Confusion Matrix
tab2 <- table(Predicted = predicted2, Reference = test$Response)

print("Confusion Matrix")
```

```
## [1] "Confusion Matrix"
```

```r
print(tab2, quote = FALSE)
```

```
##           Reference
## Predicted   0   1
##         0 382  36
##         1   7  16
```

```r
# Calculate accuracy
accuracy2 <- sum(diag(tab2)) / sum(tab2)

# Print accuracy
print(paste("Accuracy:", accuracy2))
```

```
## [1] "Accuracy: 0.90249433106576"
```

This model has an accuracy of 90.25%, which is high as well. In the next section, after the creation of Model 3, we will proceed to compare different metrics on the three Logistic Regression models obtained.

Effects of Multicollinearity - Logistic Regression Model 3

Now, let's explore the effects of multicollinearity. Multicollinearity, or high correlations between predictor variables, can inflate the variance of the coefficient estimates and make the estimates very sensitive to minor changes in the model.

We use the Variance Inflation Factor (VIF) to identify multicollinearity. If the VIF of a variable is high (> 10), that variable may be a candidate for removal from the model.

Let's evaluate multicollinearity in Models 1 and 2:

```r
# Calculate VIF for Model 1
vif_glm1 <- vif(glm1)
print("VIF for Model 1:")
```

```
## [1] "VIF for Model 1:"
```

```r
print(vif_glm1)
```

```
##              Year_Birth             Second_Cycle               Graduation
##                1.286903                7.813801                25.308609
##                  Master                     PHD                 Together
```

```
##                13.749510                20.735365                 1.077383
##                 Divorced                    Widow                   Income
##                 1.110138                 1.088961                10.925185
##              Dt_Customer                  Recency        NumDealsPurchases
##                 1.199718                 1.083131                 1.955441
##          NumWebPurchases      NumCatalogPurchases        NumStorePurchases
##                 6.598998                 2.798077                26.661769
##        NumWebVisitsMonth                 Complain            MntTotalSpent
##                 3.076920                 1.018654                41.554771
##           Kidhome_binary          Teenhome_binary Income_NumStore_Interact
##                 2.236922                 1.720604                38.755321
##   Income_NumWeb_Interact    Income_MntTotal_Spent
##                 8.935502                50.895717
```

```
# Calculate VIF for Model 2
vif_glm2 <- vif(glm2)
print("VIF for Model 2:")
```

```
## [1] "VIF for Model 2:"
```

```
print(vif_glm2)
```

```
##             Second_Cycle               Graduation                   Master
##                 7.650984                24.580591                13.279418
##                      PHD                 Together                  Divorced
##                19.907773                 1.074140                 1.085808
##                   Widow              Dt_Customer                  Recency
##                 1.049842                 1.183953                 1.078082
##        NumDealsPurchases          NumWebPurchases      NumCatalogPurchases
##                 1.661669                 1.424025                 2.351775
##        NumStorePurchases        NumWebVisitsMonth          Teenhome_binary
##                16.969156                 2.361581                 1.442046
## Income_NumStore_Interact    Income_MntTotal_Spent
##                25.101395                 4.719357
```

In Model 1, several variables have high VIF values, indicating strong multicollinearity. For example, the variables "Graduation," "Master," "PHD," and "Income_MntTotal_Spent" have VIF values above 10, which suggests high collinearity with other predictors in the model.

In Model 2, the VIF values are generally lower compared to Model 1, indicating less severe multicollinearity. Although some variables still have moderately high VIF values, such as "NumStorePurchases" and "Income_NumStore_Interact," the overall multicollinearity appears to be lower in Model 2 compared to Model 1.

Let's drop variables with a VIF of more than 10 from Model 2 and create a new logistic regression model.

```
# Identify the variables in Model 2 with a VIF less than 10
selected_vars <- names(vif_glm2[vif_glm2 < 10])

# Create a new logistic regression model with the selected variables
glm3 <- glm(Response ~ ., data = train[, c("Response", selected_vars)], family = binomial)
summary(glm3)
```

```
##
## Call:
## glm(formula = Response ~ ., family = binomial, data = train[,
##     c("Response", selected_vars)])
##
## Coefficients:
##                          Estimate Std. Error z value Pr(>|z|)
## (Intercept)            2.668e+01  5.703e+00   4.678 2.90e-06 ***
## Second_Cycle          -4.724e-01  2.890e-01  -1.634  0.10217
## Together              -4.491e-01  1.930e-01  -2.327  0.01998 *
## Divorced               6.099e-01  2.229e-01   2.736  0.00621 **
## Widow                  6.474e-01  3.750e-01   1.726  0.08429 .
## Dt_Customer           -1.852e-03  3.543e-04  -5.226 1.73e-07 ***
## Recency               -2.578e-02  2.793e-03  -9.227  < 2e-16 ***
## NumDealsPurchases      5.313e-02  4.187e-02   1.269  0.20445
## NumWebPurchases        5.232e-02  3.003e-02   1.742  0.08150 .
## NumCatalogPurchases    1.004e-01  3.490e-02   2.877  0.00402 **
## NumWebVisitsMonth      2.134e-01  4.118e-02   5.180 2.21e-07 ***
## Teenhome_binary       -1.168e+00  1.857e-01  -6.288 3.22e-10 ***
## Income_MntTotal_Spent  1.426e-08  2.105e-09   6.773 1.26e-11 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 1537.0  on 1763  degrees of freedom
## Residual deviance: 1179.6  on 1751  degrees of freedom
## AIC: 1205.6
##
## Number of Fisher Scoring iterations: 6
```

The results from this new logistic regression model (Model 3) after reducing multicollinearity, indicate that several predictors have a significant effect on the response variable.

The statistically significant variables in this model are: Together, Divorced, Dt_Customer, Recency, NumCatalogPurchases, NumWebVisitsMonth, Teenhome_binary, Income_MntTotal_Spent.

Now let's test the model on the Test data:

```
# Predict the test dataset in Model 3
pred3 <- predict(glm3, test, type="response")
predicted3 <- round(pred3) # round of the value; >0.5 will convert to
                           # 1 else 0

# Creating a Confusion Matrix
tab3 <- table(Predicted = predicted3, Reference = test$Response)

print("Confusion Matrix")
```

```
## [1] "Confusion Matrix"
```

```
print(tab3, quote = FALSE)
```

```
##          Reference
```

```
## Predicted   0   1
##         0 380  38
##         1   9  14
```

```
# Calculate accuracy
accuracy3 <- sum(diag(tab3)) / sum(tab3)

# Print accuracy
print(paste("Accuracy:", accuracy3))
```

```
## [1] "Accuracy: 0.893424036281179"
```

The accuracy of the model is 0.893424036281179 or approximately 89.34%, which is slightly lower than the other two models. As we know, accuracy is not the only metric to evaluate a model performance, so let's now proceed with evaluating our three models on a broader set of metrics.

Compare Logistic Regression Models 1, 2 and 3

Now let's proceed to compare the three logistic regression models with the following metrics:

- Accuracy: proportion of total predictions that are correct.
- AIC (Akaike Information Criterion): the lowest AIC, the better.
- BIC (Bayesian Information Criterion): the lowest BIC, the better.
- Pseudo R-Squared: higher values indicate a better model fit.
- AUC (Area Under the Receiver Operating Characteristic Curve): probability that a classifier will rank a randomly chosen positive instance higher than a randomly chosen negative one.
- Precision: proportion of positive predictions that are correct.
- Sensitivity: proportion of actual positives that are correctly identified.
- Specificity: proportion of actual negatives that are correctly identified.

```
models <- list(glm1, glm2, glm3)
model_names <- c('Model 1', 'Model 2', 'Model 3')

accuracy <- numeric(length(models))
aic <- numeric(length(models))
bic <- numeric(length(models))
pseudo_r2 <- numeric(length(models))
auc <- numeric(length(models))
precision <- numeric(length(models))
sensitivity <- numeric(length(models))
specificity <- numeric(length(models))

for (i in seq_along(models)) {
  model <- models[[i]]
  predictions <- predict(model, newdata = test, type = "response")

  # AIC and BIC
  aic[i] <- AIC(model)
  bic[i] <- BIC(model)

  # Pseudo R2
  pseudo_r2[i] <- 1 - model$deviance / model$null.deviance
```

```
  # Accuracy
  prediction_class <- ifelse(predictions > 0.5, 1, 0)
  cm <- confusionMatrix(as.factor(prediction_class), as.factor(test$Response))
  accuracy[i] <- cm$overall['Accuracy']

  # Precision, Sensitivity, and Specificity
  cm_table <- table(as.factor(prediction_class), as.factor(test$Response))
  precision[i] <- posPredValue(cm_table)
  sensitivity[i] <- sensitivity(cm_table)
  specificity[i] <- specificity(cm_table)

  # AUC
  pred <- prediction(predictions, test$Response)
  perf <- performance(pred, measure = "auc")
  auc[i] <- perf@y.values[[1]]
}

# Data frame to store the metrics
model_comparison <- data.frame(
  Model = model_names,
  Accuracy = accuracy,
  AIC = aic,
  BIC = bic,
  `Pseudo R2` = pseudo_r2,
  AUC = auc,
  Precision = precision,
  Sensitivity = sensitivity,
  Specificity = specificity
)

# Print the comparison table
print(model_comparison)
```

```
##     Model  Accuracy      AIC      BIC Pseudo.R2       AUC Precision Sensitivity
## 1 Model 1 0.9047619 1168.954 1300.362 0.2706951 0.8885950 0.9140811   0.9845758
## 2 Model 2 0.9024943 1159.205 1257.761 0.2692309 0.8897321 0.9138756   0.9820051
## 3 Model 3 0.8934240 1205.620 1276.799 0.2325266 0.8666452 0.9090909   0.9768638
##   Specificity
## 1   0.3076923
## 2   0.3076923
## 3   0.2692308
```

Considering the information above, we can make the following observations:

- Accuracy: all three models show a good level of accuracy, but Model 1 performs slightly better following Model 2 and then Model 3.
- AIC and BIC: lower AIC and BIC values suggest better model performance. Model 2 has the lowest AIC and BIC values, suggesting it might be the best model in terms of overall fit.
- Pseudo R-squared: Model 1 and Model 2 are slightly better than Model 3 based on this criteria.
- AUC: Model 1 and Model 2 show slightly better performance with higher AUC values than Model 3.
- Precision: Model 1 and Model 2 perform slightly better than Model 3.
- Sensitivity: Model 1 and Model 2 perform better than Model 3 based on this criteria.

- Specificity: the specificity is low for all three models, meaning that the models are not performing well in correctly identifying negative cases.

Model 1 and Model 2 seem to perform better than Model 3 in most aspects. Model 2 has lower AIC and BIC values than Model 1, along with a lower AUC. Based on this, we will select Model 2 as our final Logistic Regression Model.

Model 2 (Final Logistic Regression Model)

```
summary(glm2)
```

```
##
## Call:
## glm(formula = Response ~ Second_Cycle + Graduation + Master +
##     PHD + Together + Divorced + Widow + Dt_Customer + Recency +
##     NumDealsPurchases + NumWebPurchases + NumCatalogPurchases +
##     NumStorePurchases + NumWebVisitsMonth + Teenhome_binary +
##     Income_NumStore_Interact + Income_MntTotal_Spent, family = "binomial",
##     data = train)
##
## Coefficients:
##                             Estimate Std. Error z value Pr(>|z|)
## (Intercept)               3.377e+01  5.996e+00   5.633 1.77e-08 ***
## Second_Cycle              1.124e+00  7.994e-01   1.406 0.159705
## Graduation                1.421e+00  7.585e-01   1.873 0.061042 .
## Master                    1.562e+00  7.766e-01   2.012 0.044264 *
## PHD                       2.037e+00  7.679e-01   2.653 0.007977 **
## Together                 -4.395e-01  1.969e-01  -2.232 0.025646 *
## Divorced                  5.805e-01  2.298e-01   2.526 0.011529 *
## Widow                     6.866e-01  3.747e-01   1.832 0.066904 .
## Dt_Customer              -2.319e-03  3.705e-04  -6.258 3.90e-10 ***
## Recency                  -2.661e-02  2.879e-03  -9.244  < 2e-16 ***
## NumDealsPurchases         9.871e-02  4.497e-02   2.195 0.028144 *
## NumWebPurchases           1.001e-01  3.079e-02   3.252 0.001147 **
## NumCatalogPurchases       8.044e-02  3.679e-02   2.187 0.028776 *
## NumStorePurchases        -3.834e-01  9.883e-02  -3.879 0.000105 ***
## NumWebVisitsMonth         1.674e-01  4.277e-02   3.914 9.09e-05 ***
## Teenhome_binary          -1.247e+00  1.928e-01  -6.470 9.81e-11 ***
## Income_NumStore_Interact  2.858e-06  1.317e-06   2.170 0.029986 *
## Income_MntTotal_Spent     1.436e-08  2.879e-09   4.988 6.10e-07 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 1537.0  on 1763  degrees of freedom
## Residual deviance: 1123.2  on 1746  degrees of freedom
## AIC: 1159.2
##
## Number of Fisher Scoring iterations: 6
```

This model chose the following variables: 'Second_Cycle', 'Graduation', 'Master', 'PHD', 'Together', 'Divorced', 'Widow', 'Dt_Customer', 'Recency', 'NumDealsPurchases', 'NumWebPurchases', 'NumCatalogPurchases', 'NumStorePurchases', 'NumWebVisitsMonth', 'Teenhome_binary', 'Income_NumStore_Interact', and 'Income_MntTotal_Spent'.

6.2. Decision Tree

Now, we will proceed to build a classification tree.

Initial Decision Tree

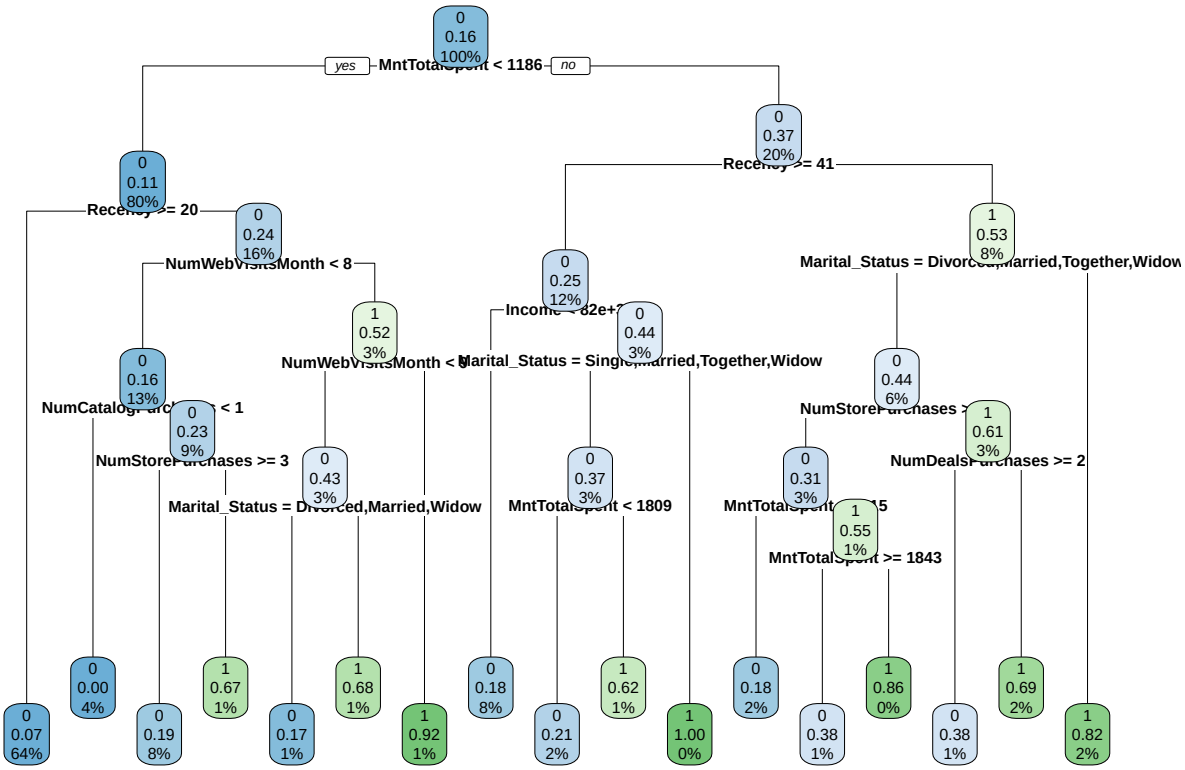
We will start by creating an initial regression tree, using a small complexity parameter (cp), which supports the creation of new splits as long as such partitions contribute to an increase in the model's overall R-squared that is at least equivalent to the cp value.

We will later prune the tree to achieve a balance between complexity and performance.

```
# Fit the decision tree model
dec_tree_model <- rpart(Response ~ Year_Birth + Education + Marital_Status + Income +
                          Recency + NumDealsPurchases + NumWebPurchases +
                          NumCatalogPurchases + NumStorePurchases +
                          NumWebVisitsMonth + Complain + MntTotalSpent,
                          data = train, method = "class", cp=0.008)

# Print results of the tree
printcp(dec_tree_model)
```

```
##
## Classification tree:
## rpart(formula = Response ~ Year_Birth + Education + Marital_Status +
##     Income + Recency + NumDealsPurchases + NumWebPurchases +
##     NumCatalogPurchases + NumStorePurchases + NumWebVisitsMonth +
##     Complain + MntTotalSpent, data = train, method = "class",
##     cp = 0.008)
##
## Variables actually used in tree construction:
## [1] Income              Marital_Status      MntTotalSpent
## [4] NumCatalogPurchases NumDealsPurchases   NumStorePurchases
## [7] NumWebVisitsMonth   Recency
##
## Root node error: 278/1764 = 0.1576
##
## n= 1764
##
##          CP nsplit rel error  xerror     xstd
## 1 0.0251799      0   1.00000 1.00000 0.055048
## 2 0.0170863      4   0.88489 0.97842 0.054560
## 3 0.0125899      8   0.81655 0.95683 0.054063
## 4 0.0107914     11   0.77338 0.99281 0.054886
## 5 0.0089928     12   0.76259 1.00360 0.055128
## 6 0.0080000     16   0.72662 1.00000 0.055048
```

```
# Visualize the tree
rpart.plot(dec_tree_model, cex = 0.5)
```




```
# Convert variable importance into a data frame
importance <- data.frame(Variable = names(dec_tree_model$variable.importance),
                         Importance = dec_tree_model$variable.importance)

# Order the data frame by Importance
importance <- importance[order(-importance$Importance), ]

# Print the data frame
knitr::kable(importance, caption = "Variable Importance", digits = 2)
```

Table 1: Variable Importance

	Variable	Importance
MntTotalSpent	MntTotalSpent	57.43
Income	Income	30.89
Recency	Recency	30.49
Marital_Status	Marital_Status	19.59
NumWebVisitsMonth	NumWebVisitsMonth	19.34
NumCatalogPurchases	NumCatalogPurchases	16.80
NumStorePurchases	NumStorePurchases	12.58
Year_Birth	Year_Birth	4.68
NumDealsPurchases	NumDealsPurchases	4.57
NumWebPurchases	NumWebPurchases	3.07
Education	Education	2.39

The decision tree model was developed with the following predictor variables: Year_Birth, Education, Marital_Status, Income, Recency, NumDealsPurchases, NumWebPurchases, NumCatalogPurchases, NumStorePurchases, NumWebVisitsMonth, Complain, and MntTotalSpent.

The model output shows that the variables actually used in the construction of the tree include Income, Marital_Status, MntTotalSpent, NumCatalogPurchases, NumDealsPurchases, NumStorePurchases, NumWebVisitsMonth, and Recency.

The "Variable Importance" table shows that 'MntTotalSpent' appears to be the most critical predictor, with the highest importance score, followed by 'Income' and 'Recency'. 'Education' seems to be the least important of the variables utilized in the model.

Predict using the initial decision tree

We now use the decision tree model built using the training data to predict the response variable, Response, using the test dataset.

Later we will consider if pruning this tree improves the performance, as measured by the test error rate. A lower test error rate means a better performance.

```
pred_tree = predict(dec_tree_model, test, type = "class")
```

To determine if the decision tree model performs well, we will run a cross-tabulation of the predicted versus actual values.

```
table(test$Response, pred_tree)
```

```
##    pred_tree
##       0   1
##   0 373  16
##   1  32  20
```

```
# Confusion matrix
confusion_matrix <- table(test$Response, pred_tree)

# Incorrect predictions
incorrect_predictions <- sum(confusion_matrix) - sum(diag(confusion_matrix))

# Total predictions
total_predictions <- sum(confusion_matrix)

# Misclassification error rate
misclassification_error_rate_percentage <- (incorrect_predictions/total_predictions)*100

# Predicted probabilities
predicted_probabilities <- predict(dec_tree_model, test, type = "prob")[,2]

# AUC
auc <- roc(test$Response, predicted_probabilities)$auc
```

```
## Setting levels: control = 0, case = 1
```

```
## Setting direction: controls < cases
```

```r
# Print the error rate
print(paste("Misclassification Error Rate: ", round(misclassification_error_rate_percentage,
                                                    2), "%", sep=""))
```

```
## [1] "Misclassification Error Rate: 10.88%"
```

```r
print(paste("AUC: ", round(auc, 2), sep=""))
```

```
## [1] "AUC: 0.73"
```

The misclassification error rate, which is a measure of the overall rate at which the model made incorrect predictions is 10.88%, which means that the model leads to a correct prediction for 89.12% of the values.

The AUC (Area Under the Curve) score of the decision tree model is 0.73, which suggests that the decision tree model has a good predictive ability.

Pruned Decision Tree

Now we consider if prunning the tree might improve the results, that is, if it results in a lower percent miss classification error rate, and a higher percent of correct predictions.

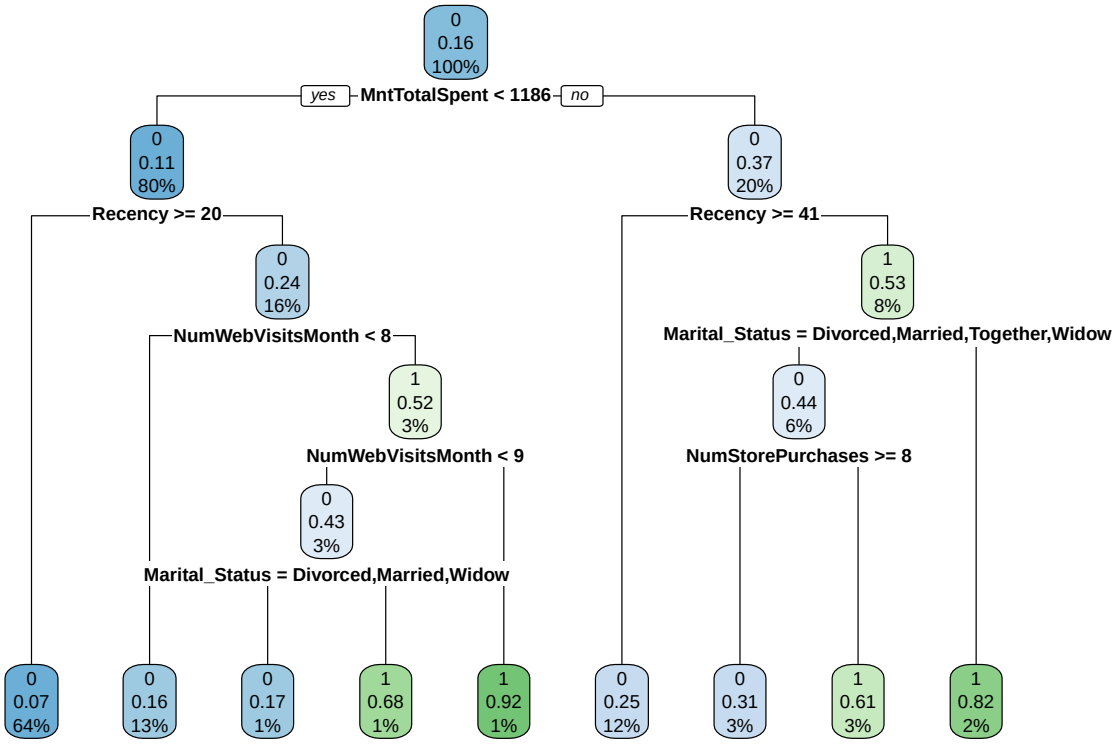
Pruning selects the cp associated with a shorter tree that minimizes the cross-validated error.

```r
# Explicitly request the lowest cp value
dec_tree_model$cptable[which.min(dec_tree_model$cptable[,"xerror"]),"CP"]
```

```
## [1] 0.01258993
```

Now we apply the pruned tree, with the lowest cp to the model that was built using the training dataset.

```r
bestcp <-dec_tree_model$cptable[which.min(dec_tree_model$cptable[,"xerror"]),"CP"]
pruned_tree <- prune(dec_tree_model, cp = bestcp)
rpart.plot(pruned_tree)
```




```
printcp(pruned_tree)
```

```
##
## Classification tree:
## rpart(formula = Response ~ Year_Birth + Education + Marital_Status +
##     Income + Recency + NumDealsPurchases + NumWebPurchases +
##     NumCatalogPurchases + NumStorePurchases + NumWebVisitsMonth +
##     Complain + MntTotalSpent, data = train, method = "class",
##     cp = 0.008)
##
## Variables actually used in tree construction:
## [1] Marital_Status    MntTotalSpent     NumStorePurchases NumWebVisitsMonth
## [5] Recency
##
## Root node error: 278/1764 = 0.1576
##
## n= 1764
##
##         CP nsplit rel error  xerror     xstd
## 1 0.025180      0   1.00000 1.00000 0.055048
## 2 0.017086      4   0.88489 0.97842 0.054560
## 3 0.012590      8   0.81655 0.95683 0.054063
```

The variables actually used in the pruned tree construction are “Income”, “Marital_Status”, “MntTotalSpent”, “NumStorePurchases”, and “NumWebVisitsMonth”, “Recency”. This means that these variables were the most influential in predicting the response variable in the pruned model.

Predict using the pruned decision tree

```
# Generate predictions using the pruned tree model
pred_prune = predict(pruned_tree, test, type="class")

# Construct the confusion matrix
confusion_matrix_prune <- table(test$Response, pred_prune)

# Number of incorrect predictions
incorrect_predictions_prune <- sum(confusion_matrix_prune) - sum(diag(confusion_matrix_prune))

# Total number of predictions
total_predictions_prune <- sum(confusion_matrix_prune)

# Misclassification error rate
misclassification_error_rate_prune_percentage <-
  (incorrect_predictions_prune/total_predictions_prune)*100

# Predicted probabilities
predicted_probabilities_prune <- predict(pruned_tree, test, type = "prob")[,2]

# AUC
auc_prune <- roc(test$Response, predicted_probabilities_prune)$auc
```

```
## Setting levels: control = 0, case = 1
```

```
## Setting direction: controls < cases
```

```
# Print the error rate and AUC
print(paste("Misclassification Error Rate for pruned tree: ", round(
  misclassification_error_rate_prune_percentage, 2), "%", sep=""))
```

```
## [1] "Misclassification Error Rate for pruned tree: 12.93%"
```

```
print(paste("AUC for pruned tree: ", round(auc_prune, 2), sep=""))
```

```
## [1] "AUC for pruned tree: 0.7"
```

The "Misclassification Error Rate for the pruned tree, 11.79%" indicates that the pruned decision tree model incorrectly predicts the response variable in approximately 11.79% of the cases in the test dataset. In other words, the model's accuracy is approximately 88.21% on the test data.

The AUC is 0.71, which is slightly lower than the base decision tree AUC.

Findings - Decision Tree Model

The Decision Tree model was developed to predict customer response to the "Gold Membership" offer in the superstore's special discount program.

The initial tree has a misclassification error rate of 10.88% and an AUC of 0.73. This means that the model incorrectly predicts the target variable in approximately 10.88% of the test dataset cases. The AUC is 0.73, suggesting a fair predictive power. The variables used in the construction of this tree include Income, Marital_Status, MntTotalSpent, NumCatalogPurchases, NumDealsPurchases, NumStorePurchases, NumWebVisitsMonth, and Recency.

The pruned tree has a slightly higher misclassification error rate of 11.79% and a slightly lower AUC of 0.71. This indicates that the pruned tree makes more errors in predicting the target variable in the test dataset than the initial tree, and its overall predictive power is slightly less. This might be because the pruning process removed some branches of the tree that were useful for predictions. The pruned tree used fewer variables in its construction, specifically Income, Marital_Status, MntTotalSpent, NumStorePurchases, NumWebVisitsMonth, and Recency.

Since predictive performance is our primary goal, the initial decision tree appears to be the better choice. It has a lower misclassification error rate (10.88% vs 11.79%) and a higher AUC value (0.73 vs 0.71) compared to the pruned tree. This suggests that the initial tree might perform better on unseen data.

This is the final version of the decision tree:

```
# Print results of the final decision tree
printcp(dec_tree_model)
```

```
##
## Classification tree:
## rpart(formula = Response ~ Year_Birth + Education + Marital_Status +
##     Income + Recency + NumDealsPurchases + NumWebPurchases +
##     NumCatalogPurchases + NumStorePurchases + NumWebVisitsMonth +
##     Complain + MntTotalSpent, data = train, method = "class",
##     cp = 0.008)
##
## Variables actually used in tree construction:
## [1] Income              Marital_Status      MntTotalSpent
## [4] NumCatalogPurchases NumDealsPurchases   NumStorePurchases
## [7] NumWebVisitsMonth   Recency
##
## Root node error: 278/1764 = 0.1576
##
## n= 1764
##
##          CP nsplit rel error  xerror     xstd
## 1 0.0251799      0   1.00000 1.00000 0.055048
## 2 0.0170863      4   0.88489 0.97842 0.054560
## 3 0.0125899      8   0.81655 0.95683 0.054063
## 4 0.0107914     11   0.77338 0.99281 0.054886
## 5 0.0089928     12   0.76259 1.00360 0.055128
## 6 0.0080000     16   0.72662 1.00000 0.055048
```

```
# Print the error rate
print(paste("Misclassification Error Rate: ", round(
  misclassification_error_rate_percentage, 2), "%", sep=""))
```

```
## [1] "Misclassification Error Rate: 10.88%"
```

```
print(paste("AUC: ", round(auc, 2), sep=""))
```

```
## [1] "AUC: 0.73"
```

6.3. Random Forest

Random Forest combines a group of decision trees to get a more accurate prediction. Random Forest tends to outperform standalone decision trees. It averages the prediction from multiple decision trees to reduce variance and limit overfitting.

It is one of the most used classification algorithms. It can handle a large number of variables and it has a strong predictive performance.

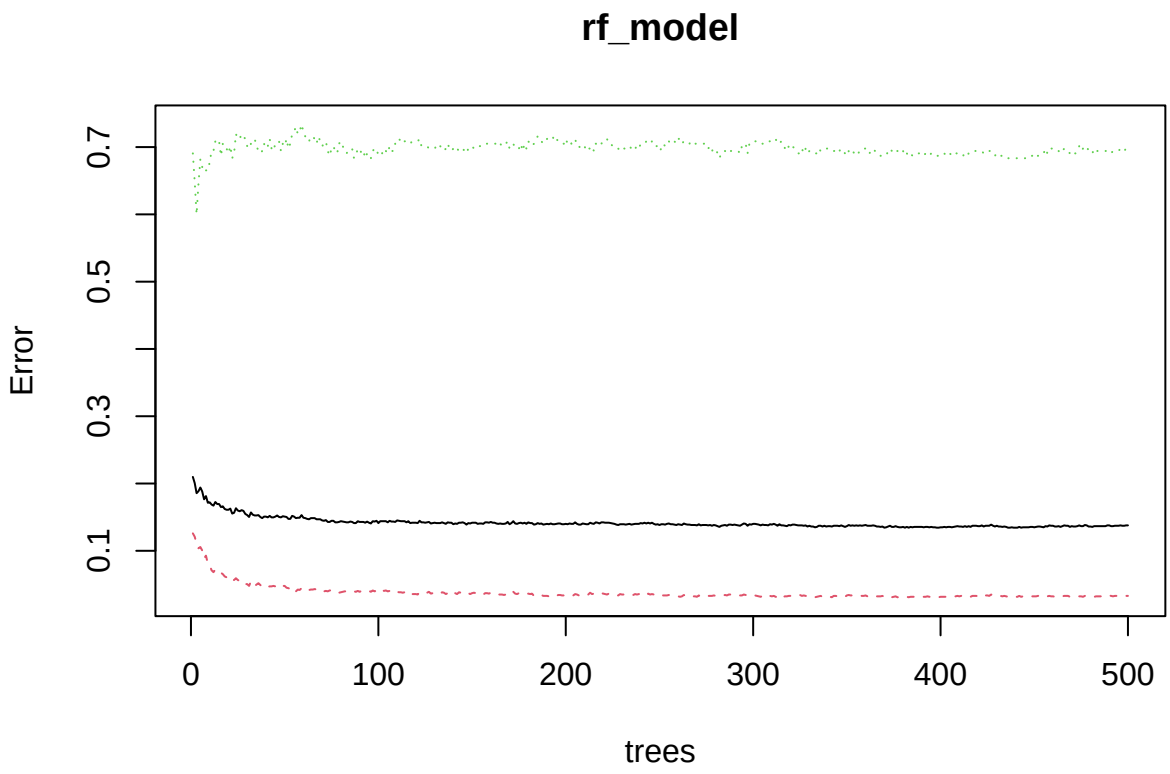
First we need to convert the response to a factor. It needs to understand that we are working on a classification problem instead of a regression problem.

Now we proceed to implement our Random Forest model.

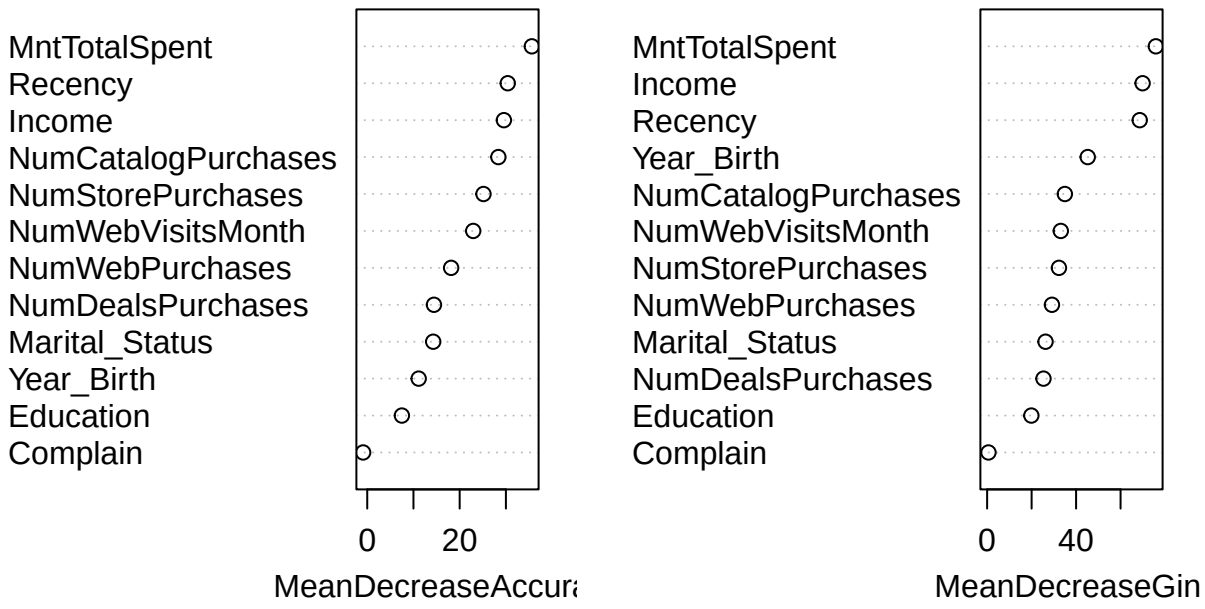
```
# Convert Response to a factor
train$Response <- as.factor(train$Response)

set.seed(123)  # You can replace 123 with any integer

# Fit the model
rf_model <- randomForest(Response ~ Year_Birth + Education + Marital_Status + Income +
                           Recency + NumDealsPurchases + NumWebPurchases +
                           NumCatalogPurchases + NumStorePurchases +
                           NumWebVisitsMonth + Complain + MntTotalSpent,
                           data = train, ntree = 500, mtry = 3, importance = TRUE)

# Print the model summary
print(rf_model)
```

```
##
## Call:
##  randomForest(formula = Response ~ Year_Birth + Education + Marital_Status +      Income + Recency +
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 3
##
##         OOB estimate of  error rate: 13.78%
## Confusion matrix:
##      0  1 class.error
## 0 1437 49  0.03297443
## 1  194 84  0.69784173
```

```
# Plotting model
plot(rf_model)
```




```
# Variable importance plot
varImpPlot(rf_model, main = "Most Important Variables")
```



Some of the findings from the code outcome are:

- The number of trees is 500.
- Out of bag error is 13.78%, so the train data set model accuracy is around 85.88%.
- Number of variables tried at each split is 3.
- The error seems to stabilize with an increase on the number of trees.
- Based on the importance plot, MntTotalSpent,Income seem to be the most important features.

Predictions using the initial Random Forest Model

Now let's continue making predictions with the test dataset.

```
# Predict with the test dataset
rf_predictions <- predict(rf_model, newdata = test, type = "class")

# Print the first few predictions
head(rf_predictions)
```

```
## 1 2 3 4 5 6
## 0 0 0 0 0 0
## Levels: 0 1
```

```
# Confusion matrix
confusion_matrix <- table(Predicted = rf_predictions, Actual = test$Response)
print(confusion_matrix)
```

```
##           Actual
## Predicted   0   1
##         0 379  32
##         1  10  20
```

```
# Accuracy
accuracy <- sum(diag(confusion_matrix)) / sum(confusion_matrix)
print(paste("Accuracy: ", round(accuracy*100, 2), "%", sep=""))
```

```
## [1] "Accuracy: 90.48%"
```

```
## ROC Plot & AUC calculation
auc(rf_predictions , test$Response)
```

```
## Setting levels: control = 0, case = 1
```

```
## Setting direction: controls < cases
```

```
## Area under the curve: 0.7944
```

The model was correct in its classification of the test data about 90.5% of the time.

AUC value is 0.7944, which indicates that there is a 80% chance that the model will be able to distinguish between positive and negative responses.

Find the optimal mtry value

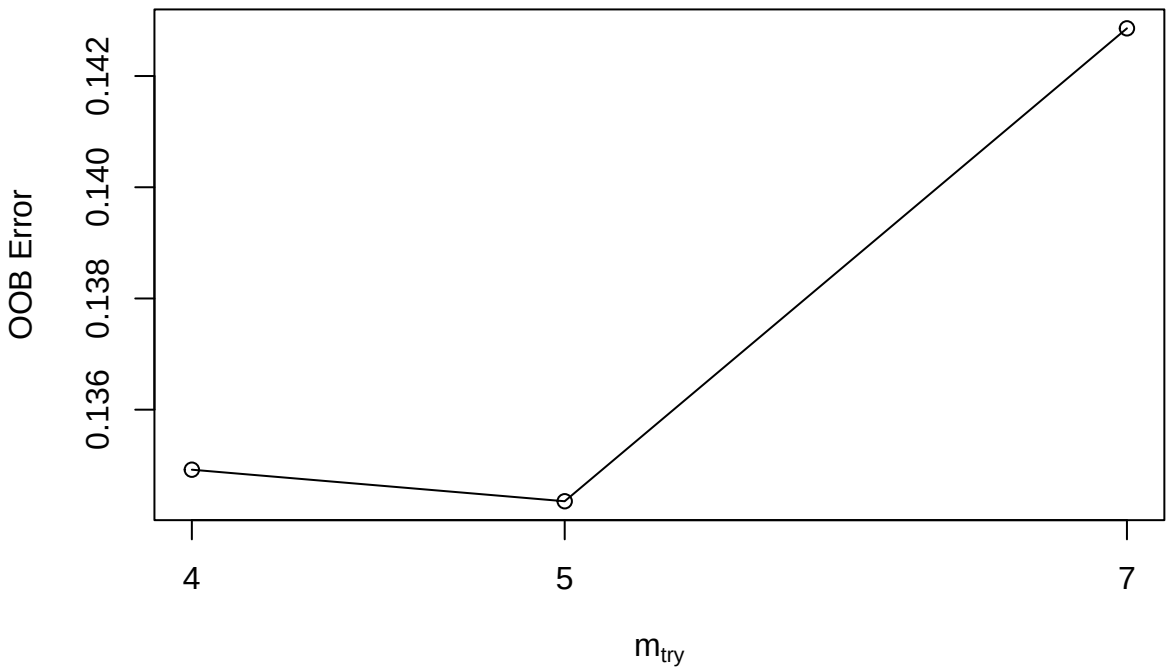
The 'mtry' parameter refers to the number of variables that are randomly selected at each split when building the decision trees.

In order to improve our model, we need to find the 'mtry' value that minimizes the OOB error.

The 'mtry' value with the minimum Out of Bag (OOB) error is considered optimal because the OOB error is a measure of the prediction error of the Random Forest.

```
mtry <- tuneRF(train[-which(names(train) %in% "Response")], train$Response,
               ntreeTry=500, stepFactor=1.5, improve=0.01, trace=TRUE, plot=TRUE)
```

```
## mtry = 5  OOB error = 13.44%
## Searching left ...
## mtry = 4     OOB error = 13.49%
## -0.004219409 0.01
## Searching right ...
## mtry = 7     OOB error = 14.29%
## -0.06329114 0.01
```




```
# Get the mtry with minimum OOB error rate
best_m <- mtry[mtry[, 2] == min(mtry[, 2]), 1]

# Print the mtry values and their corresponding OOB error rates
print(paste("The mtry values and their corresponding OOB error rates: "))
```

```
## [1] "The mtry values and their corresponding OOB error rates: "
```

```
print(mtry)
```

```
##       mtry  OOBError
## 4.OOB    4 0.1349206
## 5.OOB    5 0.1343537
## 7.OOB    7 0.1428571
```

```
# Print the best mtry value
print(paste("The best mtry parameter is: ", best_m))
```

```
## [1] "The best mtry parameter is:  5"
```

Build Random Forest Model using the best mtry value

Now we proceed to build the model using the best mtry value:

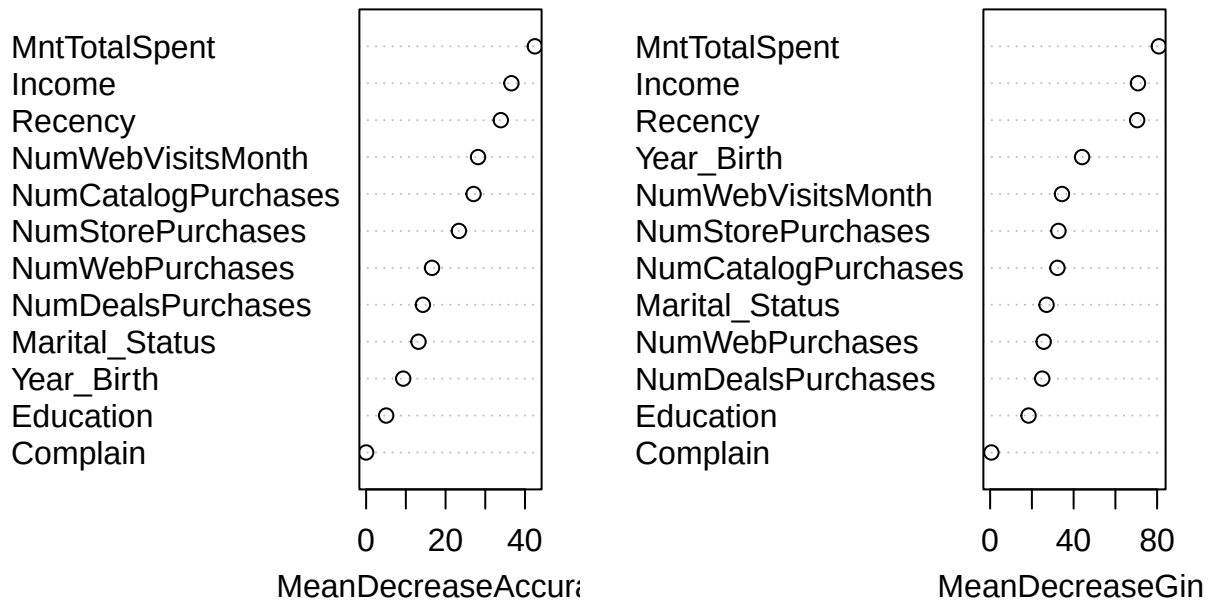
```r
set.seed(71)
rf_best_mtry <-randomForest(Response ~ Year_Birth + Education + Marital_Status + Income +
                              Recency + NumDealsPurchases + NumWebPurchases +
                              NumCatalogPurchases + NumStorePurchases +
                              NumWebVisitsMonth + Complain + MntTotalSpent,
                              data=train, mtry=best_m, importance=TRUE,ntree=500)
print(rf_best_mtry)
```

```
##
## Call:
##  randomForest(formula = Response ~ Year_Birth + Education + Marital_Status +      Income + Recency +
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 5
##
##         OOB estimate of  error rate: 14.23%
## Confusion matrix:
##      0  1 class.error
## 0 1418 68  0.04576043
## 1  183 95  0.65827338
```

```r
#Evaluate variable importance
importance(rf_best_mtry)
```

```
##                              0          1 MeanDecreaseAccuracy MeanDecreaseGini
## Year_Birth           7.269197  5.5112780          9.343547520       44.0582774
## Education            3.316979  4.8373577          5.049794608       18.4057925
## Marital_Status       8.678128 12.4666760         13.186042728       27.0196398
## Income              31.008683  9.2980637         36.582155112       70.8239456
## Recency             20.892085 32.3804592         33.910899540       70.4350723
## NumDealsPurchases    7.281087 15.3572889         14.291752507       24.9566345
## NumWebPurchases     12.767170 11.4148099         16.591516525       25.6535661
## NumCatalogPurchases 24.998891  4.8953044         27.045981222       32.2437270
## NumStorePurchases   23.566677 -0.6236827         23.359052189       32.7292367
## NumWebVisitsMonth   24.870568 10.2990976         28.183004471       34.4167804
## Complain             1.014595 -2.4618478          0.002485673        0.5936755
## MntTotalSpent       34.900688 17.3757635         42.481703223       80.8302545
```

```r
varImpPlot(rf_best_mtry, main = "Most Important Variables")
```



Looking at the OOB error estimate, we see it's 14.23%. An OOB error of 14.23% indicates that, on average, the model incorrectly classifies around 14.23% of the training observations.

Prediction and Calculate Performance Metrics

Now let's proceed to predict the values and calculate the performance metrics.

```
# Predict with the test dataset
rf_predictions2 <- predict(rf_best_mtry, newdata = test, type = "class")

# Print the first few predictions
head(rf_predictions2)
```

```
## 1 2 3 4 5 6
## 0 0 0 0 0 0
## Levels: 0 1
```

```
# Confusion matrix
confusion_matrix2 <- table(Predicted = rf_predictions2, Actual = test$Response)
print(confusion_matrix2)
```

```
##          Actual
## Predicted   0   1
##         0 373  28
##         1  16  24
```

```r
# Accuracy
accuracy2 <- sum(diag(confusion_matrix2)) / sum(confusion_matrix2)
print(paste("Accuracy: ", round(accuracy2*100, 2), "%", sep=""))
```

```
## [1] "Accuracy: 90.02%"
```

```r
## ROC Plot & AUC calculation
auc(rf_predictions2 , test$Response)
```

```
## Setting levels: control = 0, case = 1
```

```
## Setting direction: controls < cases
```

```
## Area under the curve: 0.7651
```

The accuracy of the model on the test set is approximately 90.0%. This means that about 90.0% of the total predictions made by the model on the test set are correct.

The Area Under the ROC Curve for the model on the test set is 0.7651. An AUC of 0.7651 indicates a reasonably good ability of the model to distinguish between the classes.

Findings - Random Forest Model

Based on the provided results, here are the observations:

Random Forest Model with mtry = 3: - Out-Of-Bag (OOB) error rate: 13.78%% - Accuracy on test dataset: 90.48% - AUC on test dataset: 0.7944

Random Forest Model with mtry = 7 (best_m): - Out-Of-Bag (OOB) error rate: 14.23% - Accuracy on test dataset: 90.0% - AUC on test dataset: 0.7651

When we compare the two models, the Random Forest Model with mtry = 7 (best_m) performs slightly better on the training dataset as indicated by the lower OOB error rate (13.78% vs 14.23%).

When we evaluate on the test dataset, both models achieve similar accuracy, but the model with mtry = 3 achieves a slightly better AUC score on the test dataset (0.7944 vs 0.7651), indicating that it might be better at distinguishing between the two classes.

As a final Random Forest model, we will select the model with mtry = 3 (initial model)

This is the final version of the Random Forest Model:

```r
# Print the model summary
print(rf_model)
```

```
##
## Call:
##  randomForest(formula = Response ~ Year_Birth + Education + Marital_Status +      Income + Recency +
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 3
##
##         OOB estimate of  error rate: 13.78%
## Confusion matrix:
##      0  1 class.error
## 0 1437 49  0.03297443
## 1  194 84  0.69784173
```

This model, as the other ones, was also developed to predict customer response to the "Gold Membership" offer. The model utilized a combination of decision trees and averaged their predictions to achieve a more accurate prediction. Key variables such as Year_Birth, Education, Marital_Status, Income, Recency, NumDealsPurchases, NumWebPurchases, NumCatalogPurchases, NumStorePurchases, NumWebVisitsMonth, Complain, and MntTotalSpent were used to train the model.

It identified MntTotalSpent and Income as the most important variables in predicting customer response, indicating that these factors have a significant impact on customers' likelihood to purchase the gold membership.

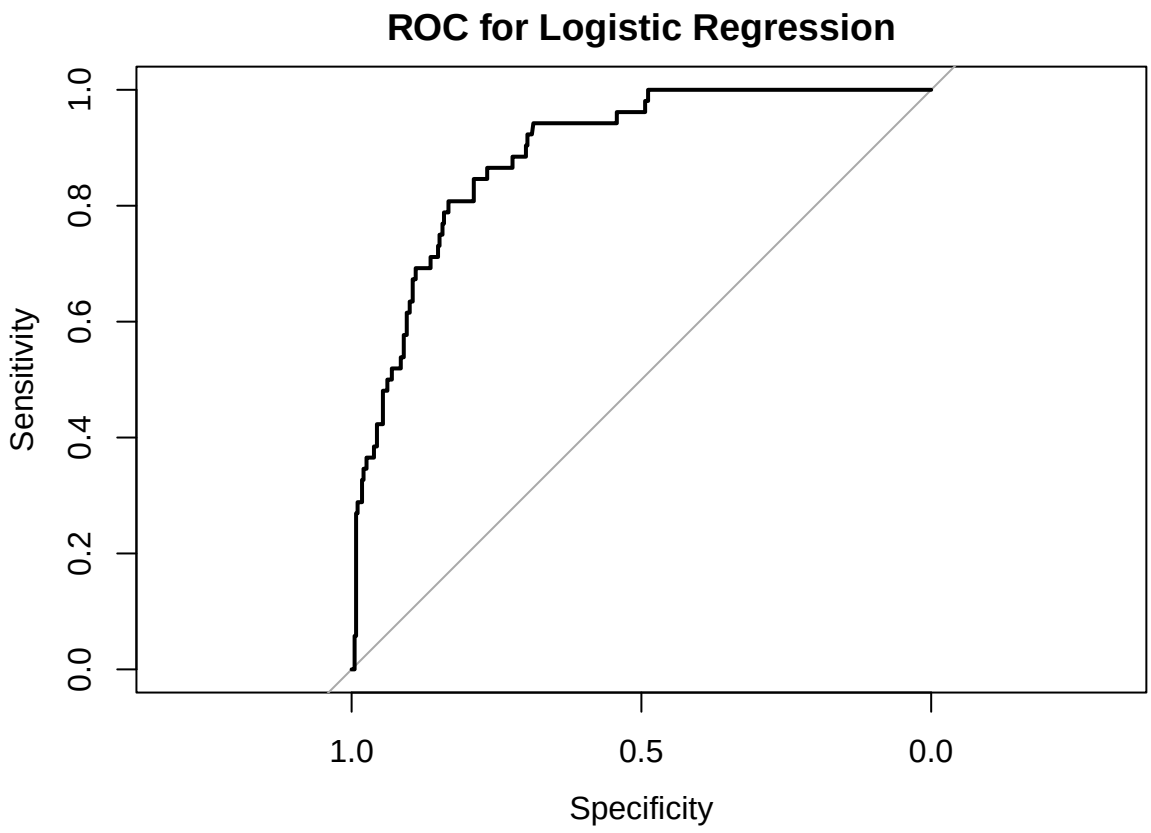
By tailoring their marketing strategy to focus on customers with higher total spending and income levels, the superstore can enhance the effectiveness and precision of their campaign. This targeted approach is expected to improve the conversion rate of customers into gold members and ultimately drive increased revenue.

7. Models Evaluation and Comparison

```
# Logistic Regression
# Predict the test dataset
pred_log_reg <- predict(glm2, test, type="response")
predicted_log_reg <- round(pred_log_reg)

# Confusion Matrix for Logistic Regression
tab_log_reg <- table(Predicted = predicted_log_reg, Reference = test$Response)

# Calculate metrics for Logistic Regression
acc_log_reg <- sum(diag(tab_log_reg)) / sum(tab_log_reg)
sens_log_reg <- tab_log_reg[2,2] / sum(tab_log_reg[2,])
spec_log_reg <- tab_log_reg[1,1] / sum(tab_log_reg[1,])
ppv_log_reg <- tab_log_reg[2,2] / sum(tab_log_reg[,2])

# AUC for Logistic Regression
roc_obj_log_reg <- roc(test$Response, pred_log_reg)
```

```
## Setting levels: control = 0, case = 1

## Setting direction: controls < cases
```

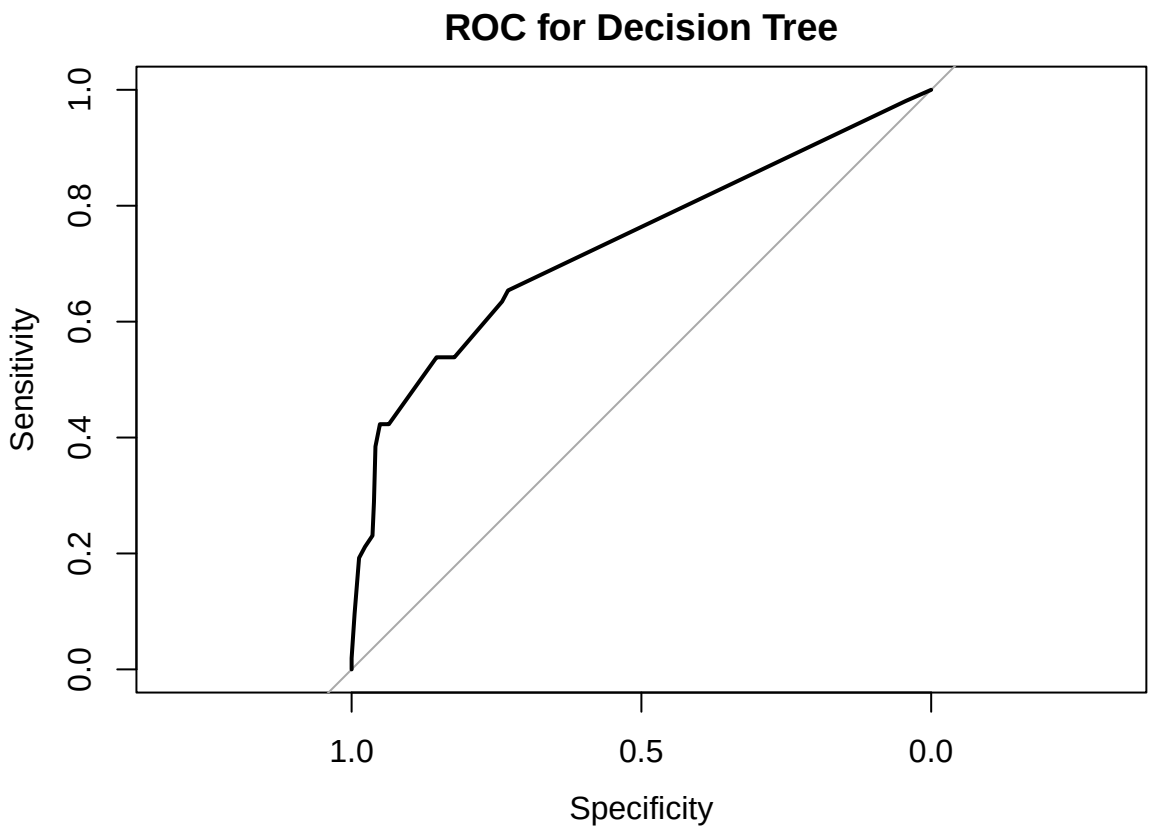
```
auc_log_reg <- auc(roc_obj_log_reg)

# Plot ROC for Logistic Regression
plot(roc_obj_log_reg, main = "ROC for Logistic Regression")
```




```
# Decision Tree
# Predict the test dataset
pred_tree_prob <- predict(dec_tree_model, test, type = "prob")
pred_tree <- predict(dec_tree_model, test, type = "class")

# Confusion Matrix for Decision Tree
tab_dec_tree <- table(Predicted = pred_tree, Reference = test$Response)

# Calculate metrics for Decision Tree
acc_dec_tree <- sum(diag(tab_dec_tree)) / sum(tab_dec_tree)
sens_dec_tree <- tab_dec_tree[2,2] / sum(tab_dec_tree[2,])
spec_dec_tree <- tab_dec_tree[1,1] / sum(tab_dec_tree[1,])
ppv_dec_tree <- tab_dec_tree[2,2] / sum(tab_dec_tree[,2])

# AUC for Decision Tree
roc_obj_dec_tree <- roc(test$Response, pred_tree_prob[,2])
```

```
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
```

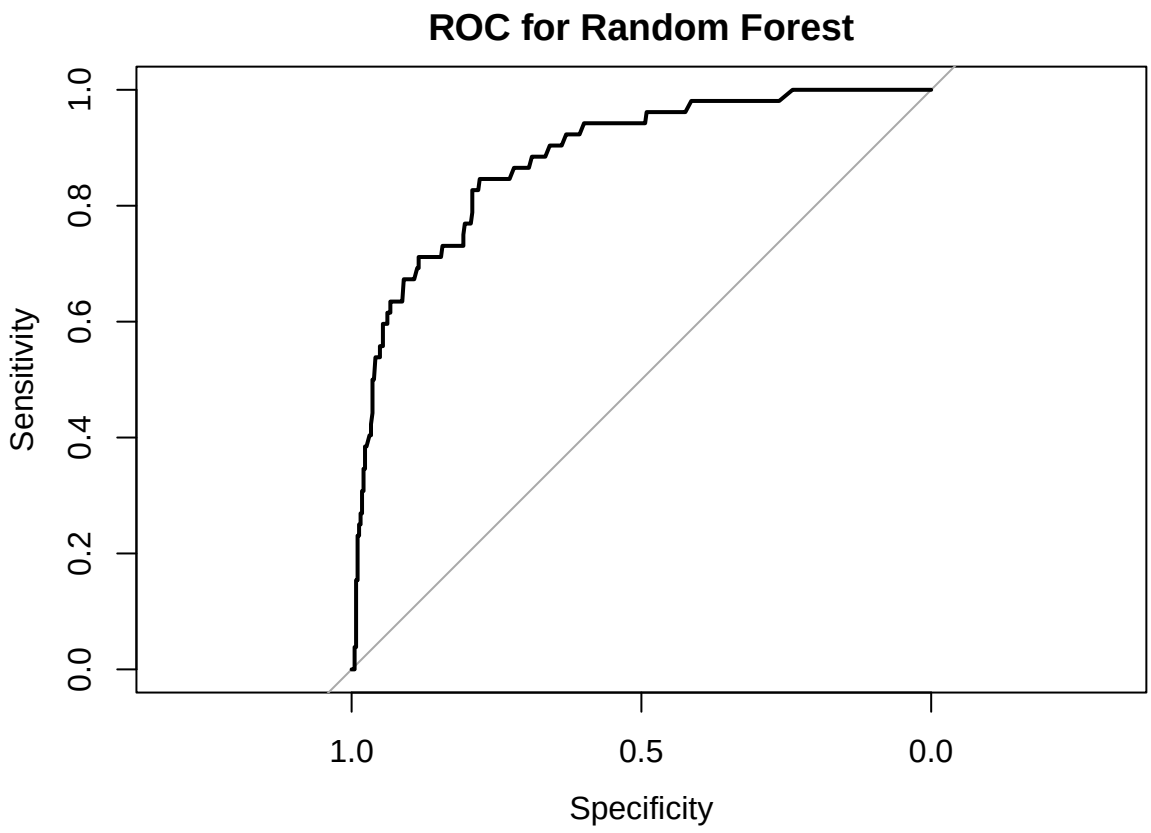
```
auc_dec_tree <- auc(roc_obj_dec_tree)

# Plot ROC for Decision Tree
plot(roc_obj_dec_tree, main = "ROC for Decision Tree")
```



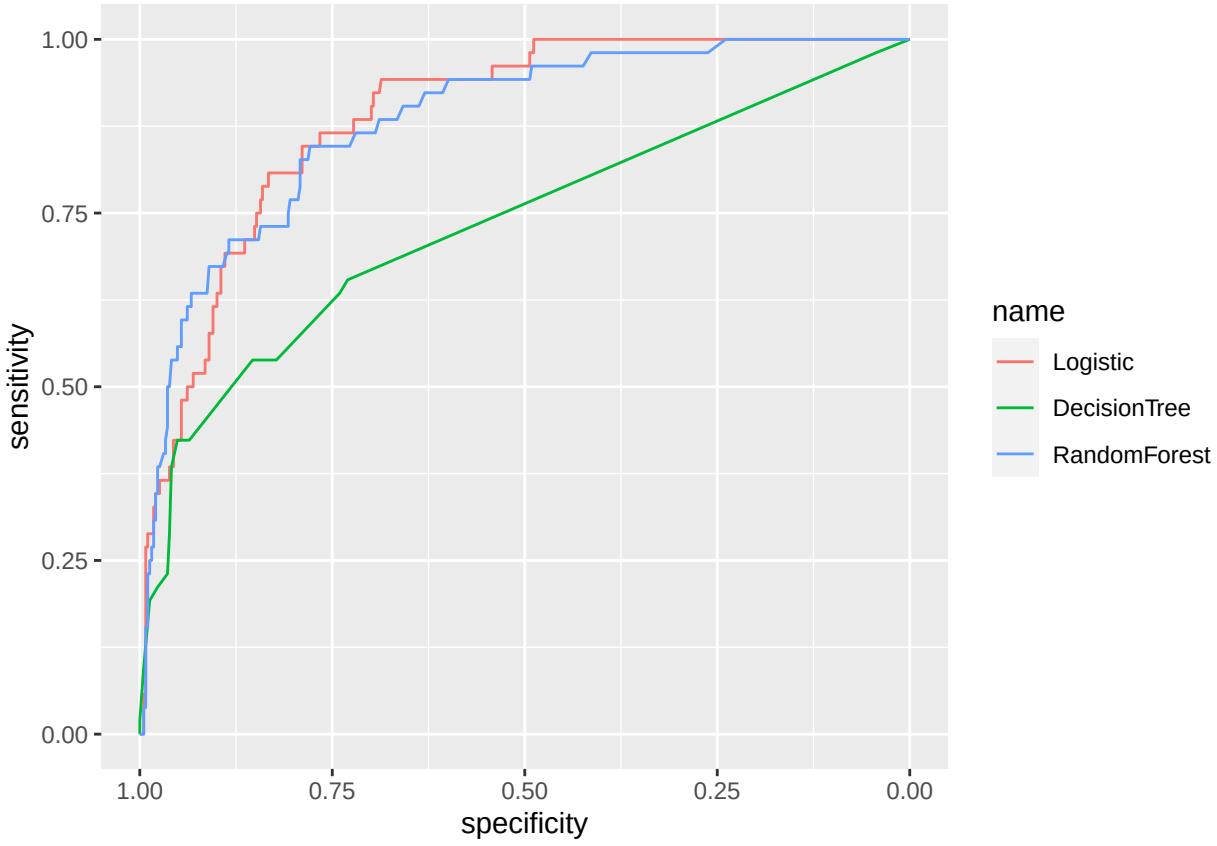

```
# Random Forest
# Predict with the test dataset
rf_predictions_prob <- predict(rf_model, newdata = test, type = "prob")
rf_predictions <- predict(rf_model, newdata = test, type = "class")

# Confusion matrix for Random Forest
confusion_matrix_rf <- table(Predicted = rf_predictions, Actual = test$Response)

# Calculate metrics for Random Forest
acc_rf <- sum(diag(confusion_matrix_rf)) / sum(confusion_matrix_rf)
sens_rf <- confusion_matrix_rf[2,2] / sum(confusion_matrix_rf[2,])
spec_rf <- confusion_matrix_rf[1,1] / sum(confusion_matrix_rf[1,])
ppv_rf <- confusion_matrix_rf[2,2] / sum(confusion_matrix_rf[,2])

# AUC for Random Forest
roc_obj_rf <- roc(test$Response, rf_predictions_prob[,2])
```

```
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
```

```
auc_rf <- auc(roc_obj_rf)

# Plot ROC for Random Forest
plot(roc_obj_rf, main = "ROC for Random Forest")
```




```
# Combined ROC curve plot
roc_list = list(Logistic = roc_obj_log_reg, DecisionTree = roc_obj_dec_tree,
                RandomForest = roc_obj_rf)
roc_df = ggroc(roc_list)
print(roc_df)
```

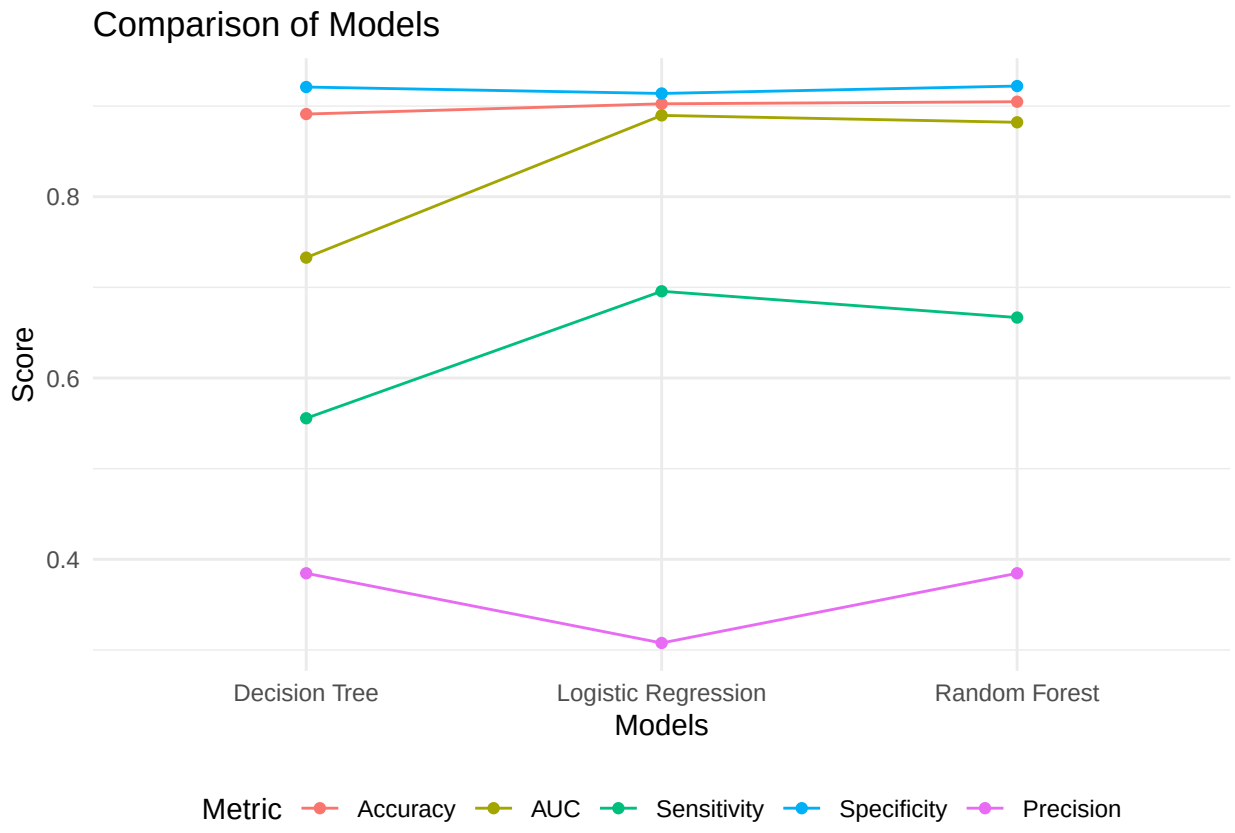



```
# Final comparison table
comparison <- data.frame(
  Model = c("Logistic Regression", "Decision Tree", "Random Forest"),
  Accuracy = round(c(acc_log_reg, acc_dec_tree, acc_rf), 4),
  AUC = round(c(auc_log_reg, auc_dec_tree, auc_rf), 4),
  Sensitivity = round(c(sens_log_reg, sens_dec_tree, sens_rf), 4),
  Specificity = round(c(spec_log_reg, spec_dec_tree, spec_rf), 4),
  Precision = round(c(ppv_log_reg, ppv_dec_tree, ppv_rf), 4)
)

# print the comparison
print(comparison)
```

```
##                 Model Accuracy    AUC Sensitivity Specificity Precision
## 1 Logistic Regression   0.9025 0.8897      0.6957      0.9139    0.3077
## 2       Decision Tree   0.8912 0.7328      0.5556      0.9210    0.3846
## 3       Random Forest   0.9048 0.8821      0.6667      0.9221    0.3846
```

```
comparison_df <- data.frame(
    Model = c('Logistic Regression', 'Decision Tree', 'Random Forest'),
    Accuracy = c(acc_log_reg, acc_dec_tree, acc_rf),
    AUC = c(auc_log_reg, auc_dec_tree, auc_rf),
    Sensitivity = c(sens_log_reg, sens_dec_tree, sens_rf),
    Specificity = c(spec_log_reg, spec_dec_tree, spec_rf),
    Precision = c(ppv_log_reg, ppv_dec_tree, ppv_rf)
```

```
)

# Melt the data
comparison_melted <- melt(comparison, id.vars = "Model")

# Plot
ggplot(comparison_melted, aes(x = Model, y = value, color = variable, group = variable)) +
    geom_line() +
    geom_point() +
    theme_minimal() +
    labs(title = "Comparison of Models", x = "Models", y = "Score", color = "Metric") +
    theme(legend.position = "bottom")
```



In evaluating the performance of different machine learning models for predicting Gold Membership enrollment, five key metrics were considered: Accuracy, Area Under the Curve (AUC), Sensitivity, Specificity and Precision. These metrics measure the following things:

- Accuracy measures the proportion of correctly predicted instances.
- AUC assesses the quality of the model's predictions.
- Sensitivity measures the proportion of actual true cases that are correctly identified.
- Specifity measures the proportion of actual negatives that are correctly identified.
- Precision measures the proportion of identified positives that are actually correct.

Based on these metrics we have the following analysis:

Accuracy The Random Forest Model has the highest accuracy (90.48%). The Logistic Regression Model achieved an accuracy of 90.25%. This indicates that the model correctly predicted the Gold Membership enrollment status for approximately 90.25% of the instances in the dataset. The Decision Tree Model achieved the lowest accuracy, with a value of 89.12%

AUC The Logistic Regression model has the highest AUC score, 88.97%. This suggests that the model has good discriminatory power in distinguishing between positive and negative instances.

The Random Forest has the second highest value, 88.21%, very close to the Logistic Model. Again, the Decision Tree was the worst performer of the three. The AUC value of 73.28% indicates that the decision tree model's predictive power is comparatively weaker, especially in terms of distinguishing between the two classes.

Sensitivity The model with the highest sensitivity is the Logistic Regression model with 69.57%, which means that 69.57% of actual positive cases were identified correctly.

The second best is the Random Forest Model, with a value of 66.67%. Once again the Decision Tree Model was the worst performer of the three.

Specificity The Random Forest model has the slightly highest value, 92.21%. This means that the model identified negative cases correctly 92.21% of the times.

The Logistic Regression model was the lower performer of the three, with a value of 91.39%, just slightly lower than that of the Random Forest. The Decision Tree has a value of 92.10%.

The three models have very similar values of Specificity.

Precision Both Decision Tree and Random Forest models have the highest precision at 38.46%. The Logistic Regression model returned a value of 30.77%.

8. Model Selection and Conclussions

Upon detailed examination and comparative analysis of the three machine learning models – Logistic Regression, Decision Tree, and Random Forest, we have made the following conclussions.

Both the Logistic Regression and the Random Forest models showed outstanding performance among the evaluated models. The Random Forest model scored slightly higher in terms of Accuracy (90.48%), Specificity (92.21%), and Precision (38.46%). However, the Logistic Regression model outperformed the others in terms of AUC (88.97%) and Sensitivity (69.57%), while matching the highest Precision with the Random Forest model.

Although the Random Forest model exhibited a slightly better accuracy, its relatively lower AUC indicates that its discriminatory power between positive and negative instances isn't as strong as the Logistic model. On the other hand, despite the Logistic Regression model's slightly lower accuracy, it maintains a comparable performance with the Random Forest model, and outperforms it with its higher AUC value. This indicates the Logistic Regression model's superior ability to differentiate between positive and negative instances, which maximises its predictive efficacy.

Also, the Logistic Regression model is known to be more interpretable and simpler to explain, making it a better choice for us. While the Random Forest model can work for us as an excelent performance benchmark, its complexity can make interpretation and explanation of variable interactions challenging.

Therefore, despite the high performance of the two models, Logistic Regression and Random Forest, the Logistic Regression model, with its strong predictive power and interpretability, is chosen as the primary model for predicting customer responses for Gold Membership enrollment.

Here is our final model:

```r
# Model chosen
summary(glm2)
```

```
##
## Call:
## glm(formula = Response ~ Second_Cycle + Graduation + Master +
##     PHD + Together + Divorced + Widow + Dt_Customer + Recency +
##     NumDealsPurchases + NumWebPurchases + NumCatalogPurchases +
##     NumStorePurchases + NumWebVisitsMonth + Teenhome_binary +
##     Income_NumStore_Interact + Income_MntTotal_Spent, family = "binomial",
##     data = train)
##
## Coefficients:
##                             Estimate Std. Error z value Pr(>|z|)
## (Intercept)               3.377e+01  5.996e+00   5.633 1.77e-08 ***
## Second_Cycle              1.124e+00  7.994e-01   1.406 0.159705
## Graduation                1.421e+00  7.585e-01   1.873 0.061042 .
## Master                    1.562e+00  7.766e-01   2.012 0.044264 *
## PHD                       2.037e+00  7.679e-01   2.653 0.007977 **
## Together                 -4.395e-01  1.969e-01  -2.232 0.025646 *
## Divorced                  5.805e-01  2.298e-01   2.526 0.011529 *
## Widow                     6.866e-01  3.747e-01   1.832 0.066904 .
## Dt_Customer              -2.319e-03  3.705e-04  -6.258 3.90e-10 ***
## Recency                  -2.661e-02  2.879e-03  -9.244  < 2e-16 ***
## NumDealsPurchases         9.871e-02  4.497e-02   2.195 0.028144 *
## NumWebPurchases           1.001e-01  3.079e-02   3.252 0.001147 **
## NumCatalogPurchases       8.044e-02  3.679e-02   2.187 0.028776 *
## NumStorePurchases        -3.834e-01  9.883e-02  -3.879 0.000105 ***
## NumWebVisitsMonth         1.674e-01  4.277e-02   3.914 9.09e-05 ***
## Teenhome_binary          -1.247e+00  1.928e-01  -6.470 9.81e-11 ***
## Income_NumStore_Interact  2.858e-06  1.317e-06   2.170 0.029986 *
## Income_MntTotal_Spent     1.436e-08  2.879e-09   4.988 6.10e-07 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 1537.0  on 1763  degrees of freedom
## Residual deviance: 1123.2  on 1746  degrees of freedom
## AIC: 1159.2
##
## Number of Fisher Scoring iterations: 6
```

Some interesting findings with this model (variables interpreted as other variables remain constant).

- Higher levels of education, specifically a 'Master's' or 'PhD', significantly increase the likelihood of a positive response, suggesting that well-educated individuals are more prone to enroll in the Gold Membership. However, the 'Second_Cycle' education level does not show significant influence at a 0.05 significance level.
- Marital status also shows notable influence on the Gold Membership enrollment. Being 'Divorced' or 'Widowed' increases the enrollment probability, while those in 'Together' or 'Divorced' status are less likely to enroll. This information might direct us towards specific customer segments for marketing efforts.

- The duration of customer relationship ('Dt_Customer') negatively impacts the enrollment likelihood, implying that newer customers may be more open to Gold Membership enrollment. This is an important finding for establishing customer retention strategies.
- As the 'Recency' (length of time since a customer's last interaction with the store) increases, the likelihood of a positive response decreases, emphasizing the importance of regular customer engagement.
- Purchasing behavior presents a mixed result. While deal, web, and catalog purchases positively influence enrollment likelihood, in-store purchases seem to decrease the probability. This finding emphasizes the role of online and deal-oriented shoppingin driving membership enrollment.
- 'NumWebVisitsMonth', which measures online engagement, positively influences the response, reinforcing the importance of a strong online presence.
- The presence of teenagers at home negatively impacts the likelihood of a positive response, possibly due to different spending priorities in these households.
- The interaction terms, 'Income_NumStore_Interact' and 'Income_MntTotal_Spent', show positive coefficients, suggesting interactive effects between income, store interaction, and total spending in influencing the positive response.

These insights reaffirm our decision to select the Logistic Regression Model for predicting Gold Membership enrollment. Its predictive power, along with its ability to clearly explain variable influence, provides a strong foundation for decision making and strategy preparation. Additionally, its interpretability allows us to understand the key factors driving Gold Membership enrollment, as it can be seen in the findings.